ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

TESIS DE GRADO

"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN ECONÓMICA Y EFICIENTE PARA VISUALIZACIÓN A ALTA RESOLUCIÓN DE IMÁGENES CIENTÍFICAS DE SIMULACIONES CLIMÁTICAS PARA SU USO EN LA ESPOL."

Previa a la obtención del título de Ingeniero en Computación especialización Sistemas de Información

PRESENTADA POR:

JORGE VICENTE SANCHEZ COMPTE

GUAYAQUIL - ECUADOR

2004

AGRADECIMIENTO

A mis papás, Jorge y Marcela, y mi hermana Catalina, por su apoyo constante e incondicional.

A Xavier, que más que un director fue un compañero y amigo en el desarrollo de esta tesis.

A mis familiares y amigos, por la presión constante para que termine esta tesis, y su ayuda en los momentos requeridos.

Al Centro de Tecnologías de Información y sus miembros, ya que sin ellos esta tesis no hubiera sido posible.

DEDICATORIA

Con cariño, a mis papás.

TRIBUNAL DE GRADO

PRESIDENTE

Ing. Miguel Yapur Auad

DIRECTOR DE TESIS

Msc. Xavier Ochoa Chehab

MIEMBROS PRINCIPALES

Dr. Boris Vintimilla Burgos

Ing. Marcelo Loor Romero

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Jorge Vicente Sánchez Compte

RESUMEN

En el Capítulo 1 se determina que la dificultad de analizar cierto tipo de datos del clima es un problema que puede ser solucionado con la utilización de un sistema de visualización de alta resolución.

También se da una definición formal de visualización, cuales son sus principales características, modos y usos. Además, se explican las diferentes tecnologías utilizadas para la visualización y proyección de imágenes, tales como proyectores digitales, pantallas de proyección, pantallas de plasma, etc.

También se hace una introducción a las librerías de software utilizadas para la generación de gráficos por computador, tales como DirectX y OpenGL.

Por último, se define formalmente que es procesamiento distribuido y sus características.

En el Capítulo 2 se analizan las soluciones posibles de hardware, las cuales son el uso de un supercomputador o de un *cluster* computadores. También se analizan las distintas soluciones de software para crear un *cluster* (Beowulf, Oscar, Clic). Además, se investigan las distintas soluciones de software para el servidor X (DMX, VNCWall), para paralelización de instrucciones gráficas (WireGL, Chromium) y para la visualización de datos (OpenDX, Vis5d+). Por último, se describe la solución escogida y su alcance.

En el Capítulo 3 se elabora el diseño lógico y físico del *cluster* de computadores. Además, se describe el diseño de los muebles necesarios para el sistema de proyección. También se describe el diseño del sistema distribuido de renderización de gráficos.

En el Capítulo 4 se hace una descripción de los pasos seguidos para la instalación y configuración del *cluster* de computadores. Además, se describe el ensamblaje de la pantalla y proyectores del sistema de proyección. También se especifican los pasos seguidos para la instalación y configuración del sistema distribuido de renderización.

En el Capítulo 5 se hace una descripción de las pruebas de funcionamiento del software de visualización (OpenDX y Vis5d+). También se hace una comparación del rendimiento del sistema con un sistema no distribuido y con otros sistemas distribuidos. Además, se muestran los resultados de pruebas del sistema en la colaboración y control remoto de aplicaciones. También se muestran los resultados de una prueba del sistema con datos reales obtenidos en la ESPOL.

INDICE GENERAL

INDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1 INTRODUCCIÓN

En este capítulo se determina que la dificultad de analizar cierto tipo de datos, entre ellos los del clima, es un problema que puede ser solucionado con la utilización de un sistema de visualización de alta resolución.

También se da una definición formal de visualización, cuales son sus principales características, modos y sus usos. Además, se explican las diferentes tecnologías utilizadas para la visualización y proyección de imágenes, tales como proyectores digitales, pantallas de proyección, pantallas de plasma, entre otros.

De igual manera, se hace una introducción a las librerías de software utilizadas para la generación de gráficos por computador, tales como DirectX y OpenGL.

Por último, se define formalmente que es procesamiento distribuido y sus características.

.

1.1 Definición del problema

Cuando se investigan distintos fenómenos, se están buscando nuevos patrones o simplemente se están analizando gran cantidad de datos, es necesario ver muchas imágenes para poder familiarizarse con el entorno, pero también se necesita que las imágenes tengan una alta resolución para poder observar los detalles en regiones pequeñas de la pantalla. Esto puede resultar problemático si se cuenta tan solo con un monitor o un proyector, ya que su tamaño limita la capacidad de mostrar muchos detalles. En cambio, una pantalla de visualización con alta resolución permite la utilización de múltiples niveles de detalle, con lo cual podemos tener gran cantidad de imágenes sin perder la calidad de las mismas.

Otro problema común al utilizar un solo monitor, es que solamente pueden trabajar o analizar los datos simultáneamente dos o tres personas. Al utilizar una pantalla, un grupo grande de investigadores y/o estudiantes puede compartir e interactuar con gran cantidad de datos y realizar simulaciones con los mismos.

En la actualidad muchas universidades y centros de investigación alrededor del mundo han desarrollado diversos sistemas de

visualización, los cuales son utilizados tanto por los científicos como por los profesores y estudiantes de dichas universidades. Algunos ejemplos de estos sistemas son el Proyecto Sandbox de la Universidad de California en Los Ángeles [53], el Sistema de Visualización de Volúmenes de la Universidad del Estado de Nueva York [58], el Centro de Visualización de la Universidad de Manchester [25], el Estudio de Visualización Científica de la NASA [49], el Ambiente Virtual Automático Cave de la Universidad de Illinois [32], entre otros.

La ESPOL no contaba con un sistema de visualización, por lo cual he decidido desarrollar una pantalla para que pueda ser utilizada tanto por los investigadores de las distintas áreas, como por los profesores y estudiantes en sus clases.

Esta tesis mostrará como se puede utilizar esta pantalla para visualizaciones científicas relacionadas con el clima, pero esto no quiere decir que el sistema de visualización no pueda ser utilizado para otros fines.

1.2 Visualización

1.2.1 Definición de visualización

De una forma concreta, visualización es el proceso de representar datos científicos abstractos con imágenes que pueden ayudar a entender el significado de estos datos. [63]

Otra manera de definir visualización es como el proceso de presentar datos de forma tal que permita el rápido entendimiento de relaciones y hallazgos que no son evidentes al revisar los datos sin procesar. [57]

Aproximadamente el cincuenta por ciento de las neuronas del cerebro están asociadas con la visión. El objetivo de la visualización es aprovechar este sistema neuronal proveyendo nuevos conocimientos científicos a través de métodos visuales. [54]

Un proverbio antiguo señala que una imagen dice más que mil palabras. Por ejemplo, si decimos “parece a un remolino. Tiene pequeños remolinos en los bordes. Tiene distintos tonos de rojo en los extremos y es mayoritariamente verde en el centro. Los remolinos más pequeños tienen el centro morado. Cada remolino tiene a su vez remolinos más pequeños. Los remolinos van en sentido de las manecillas del reloj…” Con esta descripción nos podemos dar una idea del objeto al que nos estamos refiriendo. Pero en cambio si vemos la Figura 1-1 tenemos una idea mucho más clara y precisa. [31]

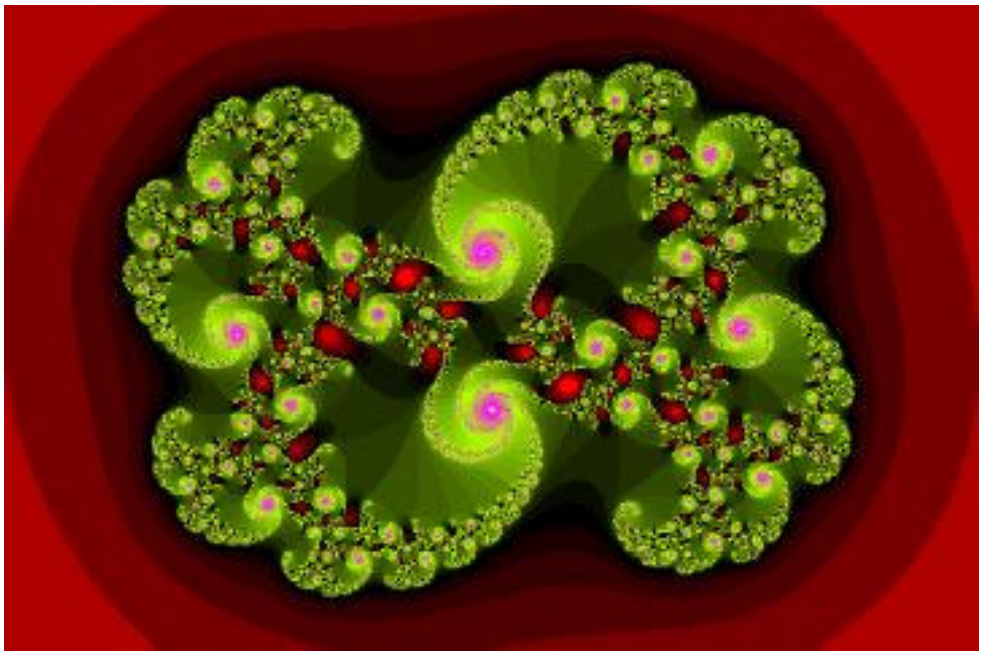


Figura 1-1 Figura de remolino [31]

En una variación moderna del proverbio podemos decir que una imagen muestra más que mil números. Por ejemplo, se pueden tener los siguientes datos:

Geometría del terreno: (10, 20, 21), (12, 13, 14), (13, 32, 12),…

(1, 2, 3), (2, 4, 5), (3, 5, 6),…

Textura del terreno: (23, 34, 54), (23, 34, 23), (45, 26, 78),…

Porcentaje de nubosidad volumétrico: 0, 0, 12, 14, 15, 15, 17, 12, 23,…

Vectores de viento: (0.2, 0.3, 0.93, 5), (0.4, 0.5, 0.76, 12),…

Al leer esta información, se pueden sacar muy pocas conclusiones sobre el estado del tiempo en una zona determinada. En cambio al ver

la Figura 1-2, resulta mucho más sencillo analizar y hasta predecir el tiempo en esa región. [31]

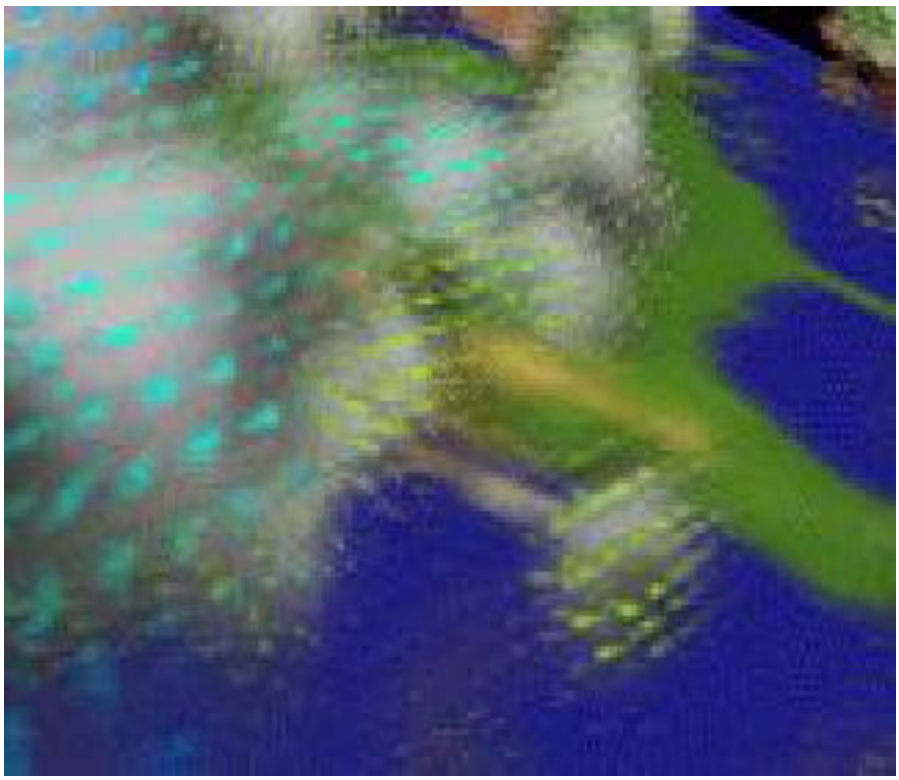


Figura 1-2 Simulación de nubosidad y viento [31]

1.2.2 Usos de la visualización

Las tecnologías de visualización permiten un conocimiento más profundo de las matemáticas, ciencias e ingeniería dentro de las universidades, negocios y agencias del gobierno. [57]

Algunas aplicaciones en las cuales es útil la visualización son:

- análisis de reservas petroleras,

- diseño asistido por computador (CAD),
- modelado molecular,
- modelado financiero,
- sistemas de información geográfica,
- simulaciones climáticas,
- manipulación de rayos x, ultrasonidos, etc.

Estas aplicaciones son útiles en muchas industrias, tales como la de hidrocarburos, automotriz, química, farmacéutica, aeronáutica, médica, etc. Además facilitan la investigación y la predicción de fenómenos. [31]

1.2.3 Características de la visualización

Las principales características de la visualización son: resolución, tamaño, brillo y tiempo de respuesta.

1.2.3.1 Resolución

El término resolución se refiere a la nitidez y claridad de la proyección de una imagen. Para poder definir la resolución, primero necesitamos entender lo que es un píxel. Píxel es la abreviación de *Picture Element*, y es un punto en una imagen gráfica. Los píxeles están tan pegados en las imágenes que parecen conectados. [62]

Cada píxel tiene un solo color para esa posición en la imagen. En la Figura 1-3 se ha hecho un acercamiento al ojo del pájaro para mostrar cada píxel individual. Cada cuadrado de la imagen agrandada es un píxel. [13]

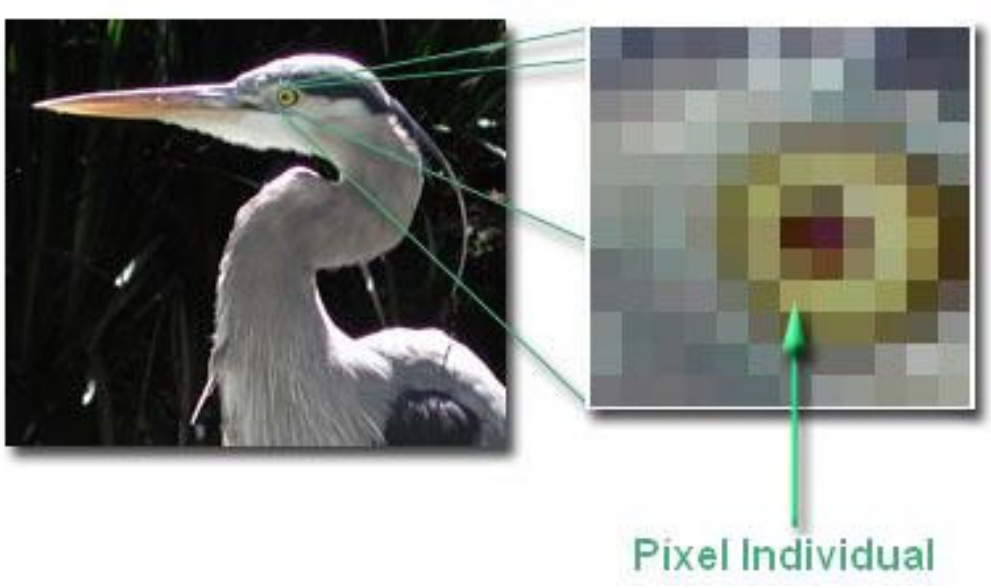


Figura 1-3 Acercamiento de una imagen para mostrar los píxeles [13]

Por definición, resolución es el número de píxeles que contiene un medio de proyección. La resolución se expresa en términos del número de píxeles en el eje horizontal y el número de píxeles en el eje vertical. [63]

La resolución también está relacionada al concepto de dpi. Dpi, es la abreviación de *dots per inch* (puntos por pulgada) y es una medida de

claridad, es decir indica la densidad de los puntos iluminados en una pantalla. [63]

Cuando se utiliza el término alta resolución, nos referimos a que existe un gran número de píxeles o puntos iluminados por pulgada cuadrada del monitor o de la proyección. [11]

Un medio de proyección tiene dos tipos de resolución principales: la resolución nativa y la resolución máxima.

1.2.3.1.1 Resolución Nativa

También llamada resolución real o verdadera, se refiere al número real de píxeles físicos en un medio de proyección. [42] Por ejemplo, si un proyector tiene una resolución nativa de 800 x 600, quiere decir que este posee 800 píxeles horizontales y 600 píxeles verticales, y por lo tanto solamente puede mostrar un máximo de 480,000 píxeles simultáneamente. [40]

La resolución nativa es muy importante porque solamente en esta resolución las imágenes se ven con la mejor definición posible. Si se utiliza una resolución mayor o menor a la nativa, las imágenes deben pasar por un proceso de compresión llamado escalamiento, en el cual se pierde información. [33]

1.2.3.1.2 Resolución Máxima

La mayoría de los proyectores aceptan una resolución mayor a su resolución nativa, esta es llamada resolución máxima. Esta resolución se logra a través de la compresión de imágenes en menos píxeles, esto quiere decir que algunos de los píxeles enviados por el computador comparten un solo píxel en la imagen proyectada. [40]

El proceso mediante el cual se comprimen las imágenes se llama escalamiento. Cabe recalcar que el escalamiento se da tanto cuando se aumenta como cuando se reduce la resolución de la imagen. Como podemos ver en la Figura 1-4, la imagen se distorsiona tanto cuando se aumenta la resolución, como cuando se disminuye. En esta figura se muestra un texto original, y el mismo texto utilizando 30% menos y 30% más píxeles. [33]

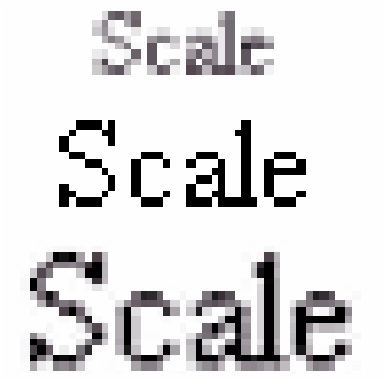


Figura 1-4 Escalamiento de una imagen [33]

La distorsión de las imágenes cuando se les cambia la resolución también depende del método que se utilice para la compresión. En la Figura 1-5 podemos ver la diferencia de cómo se ve un texto al utilizar dos tipos de compresión. La segunda línea es el resultado más común en la mayoría de los proyectores. [40]

Figura 1-5 Diferencia de dos líneas de texto con diferente compresión [40]

En la primera línea de la Figura 1-5, el método de compresión utiliza una técnica conocida como *anti-aliasing*, la cual consiste en utilizar distintos tonos de grises en los bordes de las letras o figuras para engañar al ojo y que estos bordes den la impresión de formar líneas curvas. [2]

1.2.3.2 Tamaño

El tamaño se refiere a la dimensión máxima de la pantalla utilizada para una visualización. El método más utilizado para determinar el tamaño es el de diagonal de pantalla, que consiste en medir desde

una esquina de la pantalla hasta la esquina opuesta. Por ejemplo si una pantalla tiene 9 pulgadas de alto y doce pulgadas de ancho, se dice que es una pantalla de 15 pulgadas, ya que esa es la medida de su diagonal. [47]

Los monitores tienen un tamaño fijo. En cambio, si se utilizan proyectores, el tamaño de la proyección puede variar de acuerdo a que tan cerca se ponga el proyector de la pantalla. Hay que recordar que si la resolución se mantiene constante, y aumentamos el tamaño de la proyección, los píxeles van a ser cada vez más grandes, es decir se van a tener menos puntos por cada pulgada de imagen (dpi), lo cual puede distorsionar las imágenes.

1.2.3.3 Brillo

El brillo se refiere a la luminosidad de la imagen de una visualización. Del brillo dependen las condiciones de claridad u oscuridad en las que debe estar el cuarto donde se realice la visualización.

El ANSI Lumen es la unidad de brillo definida por el American National Standards Institute, y está definida como la cantidad total de luz que, una vez que ha pasado a través del lente, llega a la pantalla. [4]

Como regla general se puede decir que mientras más lúmenes ANSI tenga el proyector que se utiliza, la imagen se verá más brillante. [17]

En la actualidad los proyectores se pueden agrupar en cuatro rangos de lúmenes ANSI:

Los proyectores con menos de 1000 lúmenes son los más baratos, pero necesitan que la habitación donde se realiza la presentación esté oscura o con muy poca luz. Además, el proyector debe estar muy cerca de la proyección. [17]

Los proyectores que tienen entre 1000 y 2000 lúmenes son un poco más caros, pero no necesitan que el cuarto esté totalmente oscuro. Este tipo de proyectores son recomendables para aulas de clase y salas de conferencia. [17]

Los proyectores que tienen de 2000 a 3000 lúmenes son más caros aún, pero permiten que se realice una proyección sin necesidad de tomar en cuenta cuanta luz tenga la habitación. De igual manera, el proyector pude encontrarse a mayor distancia de la proyección. Estos proyectores son apropiados para grandes salones de conferencia. [17]

Los proyectores con más de 3000 lúmenes son extremadamente costosos y se los utiliza para aplicaciones particulares en lugares muy grandes como auditorios, iglesias, discotecas, conciertos, etc. [17]

1.2.3.4 Tiempo de respuesta

Tiempo de respuesta es el tiempo transcurrido entre el fin de un pedido o demanda a un sistema computacional y el comienzo de la

respuesta. Por ejemplo el periodo de tiempo desde que se termina de hacer una pregunta hasta que aparece el primer carácter de la respuesta en el terminal del usuario. [63]

El tiempo de respuesta percibido, en cambio, se refiere al tiempo que el usuario siente que el computador se demora en responder. Este lapso está comprendido desde que el usuario empieza a ingresar los datos hasta que el computador termine de dar los resultados. [63]

El tiempo de respuesta normalmente se mide en segundos o alguna de sus fracciones, por ejemplo milisegundos.

Un concepto relacionado es el de latencia, que se refiere a cualquier retraso o espera que aumenta el tiempo de respuesta real o percibido. Algunos factores que contribuyen a la latencia de un sistema de computación pueden ser la diferencia de velocidad entre el microprocesador y los dispositivos de entrada o salida, el uso inadecuado de *buffers* de datos, entre otros. [63]

En el campo de la visualización, el tiempo de respuesta se refiere al tiempo que se demora un sistema en dibujar una imagen. Una forma alternativa de determinar el tiempo de respuesta de un sistema de visualización son los *fps*, que son las siglas en inglés de *frames per second*, y se refieren a la cantidad de cuadros por segundo que puede dibujar el sistema.

1.2.4 Modos de visualización

El término modo de visualización se refiere al máximo número de colores y la máxima resolución permitida por un medio de visualización, tal como un monitor o un proyector. En la Figura 1-6 podemos observar los tamaños de los modos de visualización más comunes en la actualidad.

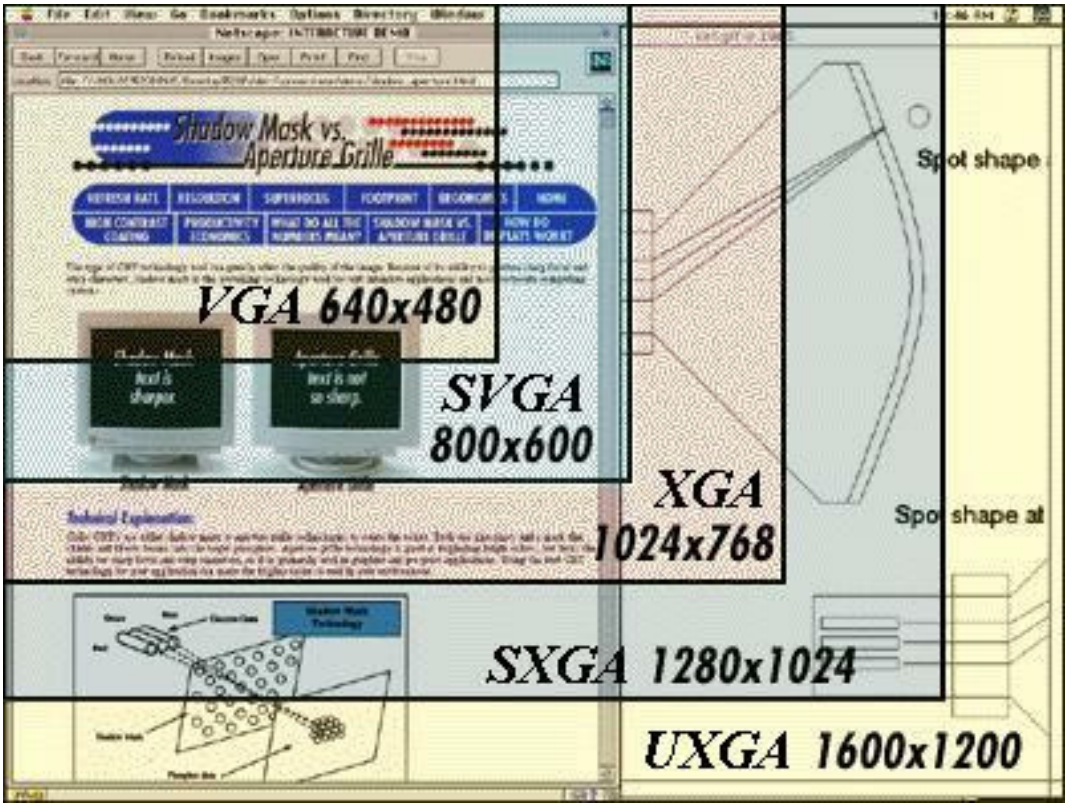


Figura 1-6 Tamaño de pantalla de los modos de visualización VGA, SVGA, XGA, SXGA, y UXGA [18]

En los años setenta, los primeros computadores personales utilizaban monitores monocromáticos. En 1981, IBM introdujo el *Color Graphics Adapter* (CGA), que permitía la renderización de 4 colores, y tenía una resolución máxima de 320 x 200. [63]

En 1984 apareció el *Enhanced Graphics Adapter* (EGA), este permitía el uso de 16 colores y ofrecía una resolución de 640 x 350. En 1987, IBM introdujo el *Video Graphics Array* (VGA), este permitía escoger entre utilizar 16 colores a 640 x 480, o 256 colores a 320 x 200. VGA se ha convertido en el estándar mínimo aceptado por los computadores personales. [63]

Luego apareció el *Super Video Graphics Array* (SVGA), el cual permitía una resolución de 800 x 600 y un a paleta de hasta 16 millones de colores (también conocido como “color verdadero”). En 1990 apareció el *Extended Graphics Array* (XGA), el cual permitía una resolución de 1024 x 768. [63]

Recientemente salieron dos nuevos modos de visualización: el *Super Extended Graphics Array* (SXGA) y el *Ultra Extended Graphics Array* (UXGA). Estos modos permiten una resolución de 1280 x 1024 y de 1600 x 1200, respectivamente. [63]

Figura 1-7 Comparación de una misma foto vista con distintos modos de visualización [18]

Como se muestra en la Figura 1-7, mientras más avanzado sea el modo de visualización, y por consiguiente más alta sea la resolución, se pueden observar fotos más grandes con un mayor nivel de detalle.

1.2.5 Visualización en tres dimensiones

Los gráficos de computador en 3D son en realidad imágenes bidimensionales que tienen una ilusión de profundidad, la cual aparece como una tercera dimensión. [66] En la Figura 1-8 podemos ver un mapa del Ecuador en dos dimensiones, en cambio en la Figura 1-9 se muestra un mapa del Ecuador con la tercera dimensión.

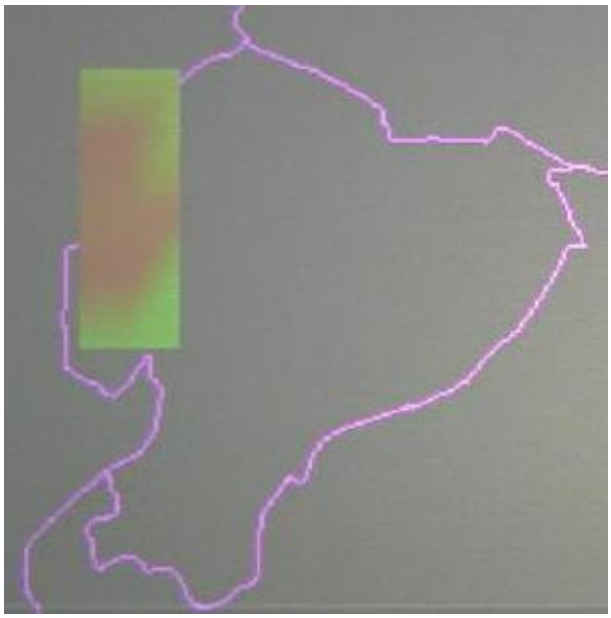


Figura 1-8 Mapa del Ecuador en dos dimensiones

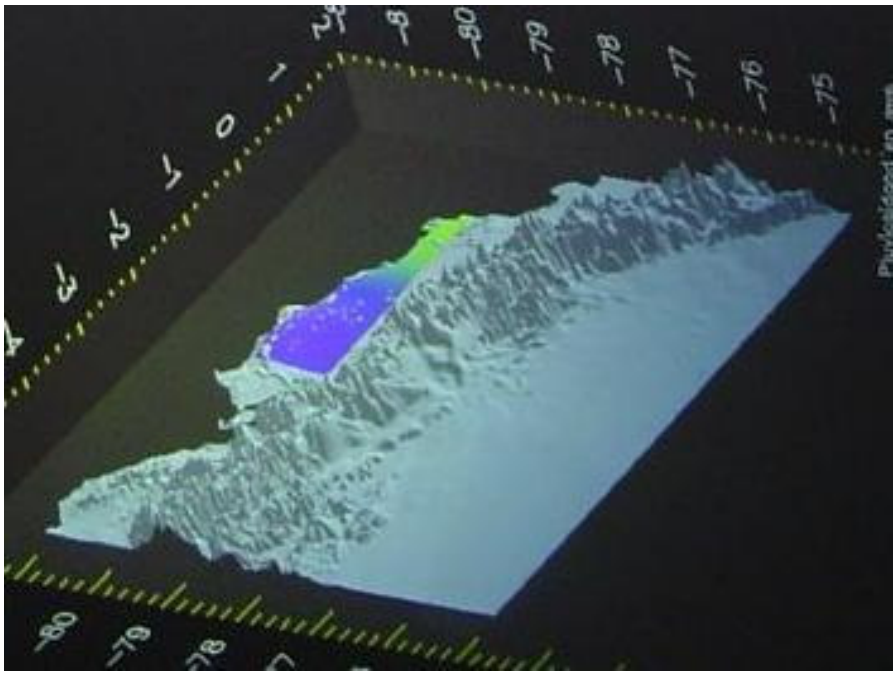


Figura 1-9 Mapa del Ecuador en tres dimensiones

Para crear imágenes en tres dimensiones se utilizan varios artificios. El primero, es la perspectiva que consiste en dibujar los objetos cercanos más grandes que los lejanos. Luego se borran las líneas y objetivos que son tapados por otros más cercanos. Por último, se oscurecen los objetos y se crea una sombra, para dar la ilusión de que existe una fuente de luz. [66]

Cuando se dibuja un objeto en tres dimensiones, este en realidad está compuesto por formas más pequeñas llamadas primitivas. Las primitivas son superficies bidimensionales tales como puntos, líneas y polígonos que al juntarse en un espacio de tres dimensiones forman objetos tridimensionales. [66]

Para poder especificar la posición de un objeto tridimensional en una pantalla se debe convertir la información de tres dimensiones en coordenadas cartesianas. Esto se conoce como proyección. Existen dos clases de proyección, la ortográfica o paralela, y la proyección con perspectiva. [66]

En la proyección ortográfica todos los objetos que tienen las mismas dimensiones son dibujados del mismo tamaño, sin importar si están más lejos o más cerca. Este tipo de proyección es utilizada principalmente para el diseño arquitectónico y el diseño asistido por computador (CAD). [66]

La proyección con perspectiva, en cambio, muestra los objetos distantes más pequeños que los objetos cercanos. Este tipo de proyección es más realista, y es utilizada principalmente para simulaciones y animaciones. [66]

Una visualización también pude ser en cuatro o cinco dimensiones. La cuarta dimensión es el tiempo, es decir una visualización en cuatro dimensiones es una animación, en la cual se muestran distintos estados a través del tiempo.

Una visualización en cinco dimensiones se refiere a las tres dimensiones espaciales más el tiempo y alguna otra variable física, tal como temperatura o presión. [55]

1.2.6 Tecnologías de visualización y proyección

En la actualidad los medios más utilizados para la presentación y visualización de imágenes a grupos grandes de personas son los proyectores digitales y las pantallas de proyección. También existen otras tecnologías que por su alto precio todavía no se utilizan de forma masiva, tales como las pantallas de plasma, OLED, entre otros.

1.2.6.1 Proyectores digitales

Proyector digital es un dispositivo que recibe una entrada de un computador o un sistema de vídeo (VHS, DVD, etc.), que contiene su propia fuente de luz, potencia y lentes, y convierte esta señal de entrada en una señal luminosa que emite hacia una pantalla de proyección. [47]

Figura 1-10 Proyectores digitales de distintas marcas [43]

Todos los proyectores digitales tienen la capacidad de ser utilizados como proyectores de retaguardia, es decir que el proyector proyecta la imagen detrás de la pantalla. Una gran ventaja de esto es que las personas no pueden caminar entre el proyector y la proyección, por lo tanto no se puede tapar la imagen. [41]

En la actualidad, para realizar la proyección, la mayoría de los proyectores digitales utilizan una de estas dos tecnologías: *Liquid Crystal Display* (LCD) o *Digital Light Processing* (DLP).

Además existe una tercera opción que es el *Cathode Ray Tube* (CRT) de 3 cañones.

1.2.6.1.1 Tecnología LCD

Los proyectores LCD utilizan tres paneles de cristal líquido, uno por cada color primario (rojo, verde y azul). Cuando la luz pasa a través de los paneles, píxeles individuales pueden ser abiertos o cerrados para permitir el paso de la luz. Esta actividad modula la luz y produce la imagen que es proyectada en la pantalla. [40]

En la Figura 1-11 se puede ver como funciona la tecnología LCD. Esta tecnología es más antigua, pero también es mucho más barata.

Otra ventaja de esta tecnología es que utiliza la luz de manera más eficiente, es decir, produce imágenes más brillantes que los proyectores DLP. Además produce colores más saturados y una imagen más puntiaguda. [40]

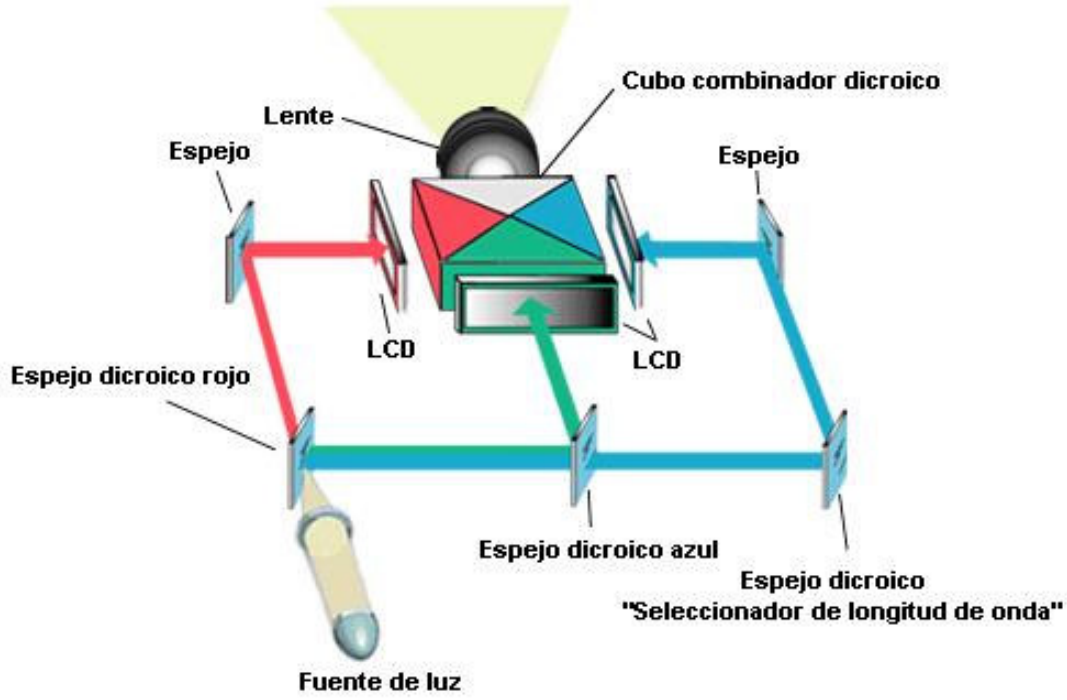


Figura 1-11 Tecnología LCD [40]

La principal desventaja de esta tecnología es que hay una gran separación de entre los píxeles, lo que produce el efecto de “tela metálica”, como se puede ver en la Figura 1-13. [40]

1.2.6.1.2 Tecnología DLP

DLP es el nombre comercial para una nueva tecnología de proyección desarrollada por *Texas Instruments* que también es conocida como tecnología de micro espejos o DMD. [47]

En esta tecnología, el proyector utiliza espejos diminutos, cada uno de los cuales representa un píxel. La luz, en lugar de pasar a través del panel, es reflejada. El color se obtiene al pasar la luz de la lámpara por una rueda que tiene filtros con los colores primarios. [63]

En la Figura 1-12 podemos ver como funciona la tecnología DLP o de micro espejos. Esta tecnología es relativamente nueva, pero tiene un precio más elevado que la anterior.

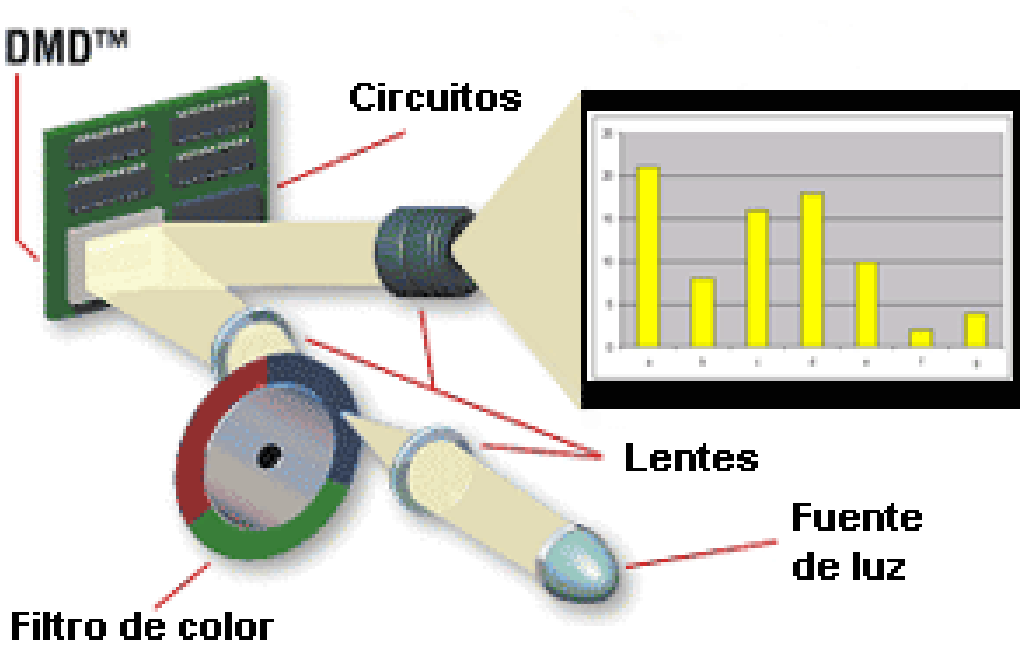


Figura 1-12 Tecnología DLP o de micro espejos [40]

La principal ventaja de esta tecnología sobre la de LCD es que los píxeles están mucho más unidos, por lo que desaparece el efecto de "tela metálica" como se puede ver en la Figura 1-13 .[40]

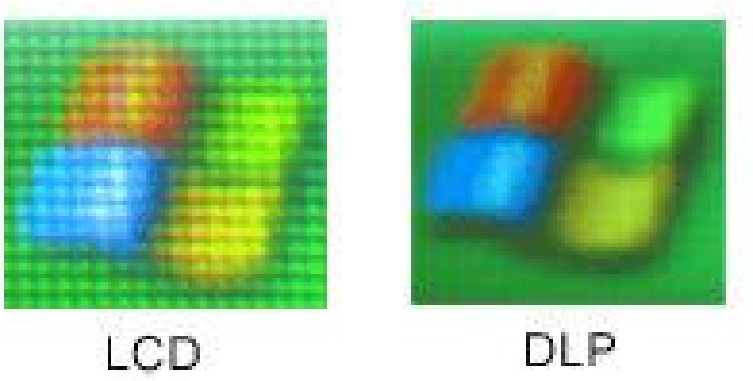


Figura 1-13 Efecto de "tela metálica" en proyectores LCD y DLP [40]

Las principales desventajas de la tecnología DLP son una menor saturación de color, y el efecto "arco iris." Este efecto solo puede ser percibido por ciertas personas, y es causado por la velocidad de la rueda en la que se encuentran los filtros con los colores primarios. [40]

1.2.6.1.3 CRT de 3 Cañones

Esta tecnología permite proyectar imágenes de mayor resolución que cualquier proyector LCD o DLP, llegando inclusive a una resolución de 2500 x 2000. De igual manera, presenta colores más ricos y

profundos. Por estas razones, esta tecnología es utilizada por proyectores para aplicaciones especiales. Estos proyectores tienen un peso elevado (hasta 200 libras) y requieren un mantenimiento constante. [47] Además, toma mucho tiempo enfocarlos y calibrarlos.

Otra desventaja de este tipo de proyectores es su alto costo. Por ejemplo el proyector *Vivid White* fabricado por *Christie Digital*, que posee una resolución nativa de 1600 x 1200, cuesta más de 50,000 dólares. [43]

1.2.6.2 Pantallas de proyección

Pantalla de proyección es una superficie especial de alta ganancia utilizada para mostrar imágenes enviadas por un proyector. [45]

Las pantallas de proyección dispersan la luz de tal manera que cada uno de los rayos provenientes del proyector es esparcido a través de la misma. Esto permite que se pueda ver la imagen proyectada en las esquinas de la pantalla y no solamente en el centro. Si una pantalla de proyección tiene una ganancia demasiado alta, la diseminación de la luz será mínima y las esquinas aparecerán oscuras y turbias, mientras que el centro se verá excesivamente brillante. [28]

Figura 1-14 Pantalla de proyección [45]

Las pantallas pueden ser de: proyección delantera y de proyección trasera.

1.2.6.2.1 Pantallas de proyección delantera

Las pantallas de proyección delantera son pantallas que reflejan la luz que reciben. Estas pantallas necesitan que la fuente de luz (el

proyector) se encuentre delante de la pantalla, como se puede ver en la Figura 1-15.

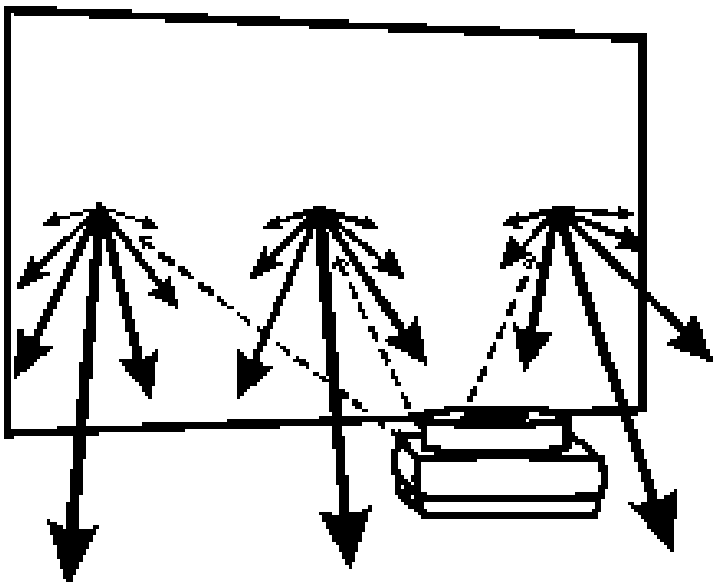


Figura 1-15 Pantalla de proyección delantera [28]

La ventaja de utilizar este tipo de pantallas es que debido a que la audiencia y el proyector se encuentran del mismo lado de la pantalla, la arquitectura del lugar donde se la utilice puede permanecer inalterada. [28]

La desventaja es que este tipo de pantalla refleja indiscriminadamente todo tipo de luz que llegue a la misma, razón por la cual la luz del proyector puede verse diluida por otras fuentes de luz, tales como focos y ventanas. [28]

1.2.6.2.2 Pantallas de proyección trasera

Las pantallas de proyección trasera son pantallas que trasmiten la luz que reciben. Estas pantallas necesitan que la fuente de luz (el proyector) se encuentre detrás de la pantalla, como se puede ver en la Figura 1-16.

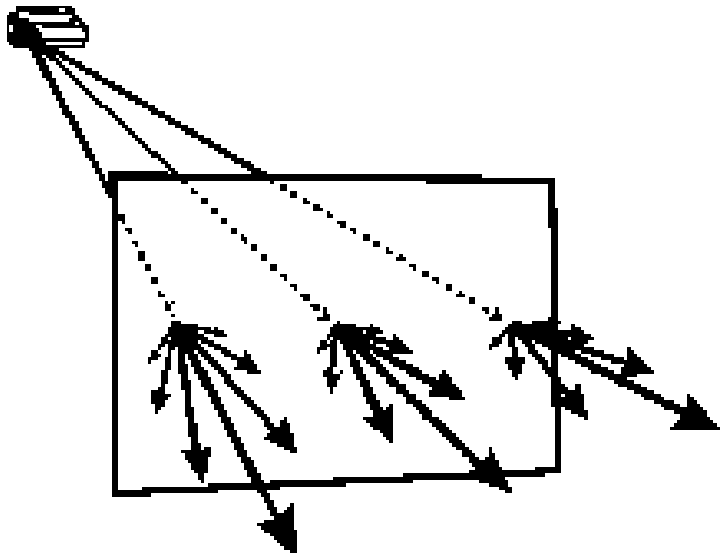


Figura 1-16 Pantalla de proyección trasera [28]

La principal ventaja de utilizar este tipo de pantallas es que, debido a que solamente transmiten la luz que se encuentra detrás de la pantalla, si existen otras fuentes de luz en frente de la pantalla (como focos, lámparas, etc.), estas no se ven reflejadas en la misma. [28]

La desventaja, en cambio es que como el proyector debe estar detrás de la pantalla, este necesita un pequeño cuarto o cabina. El tamaño

de este cuarto es directamente proporcional al tamaño de la pantalla. Además debe procurarse que este cuarto sea lo más oscuro posible para que no transmita luz no deseada. [28]

1.2.6.3 Otras tecnologías

1.2.6.3.1 Pantallas de plasma

Figura 1-17 Pantalla de plasma [18]

Una pantalla de plasma es un tipo de monitor de pantalla plana que funciona con una combinación de los gases neón y xenón encerrada

entre dos planchas selladas de vidrio. [62] Cada píxel de la pantalla es iluminado por un poquito de plasma o gas cargado, de forma parecida a un pequeño letrero de neón. [63]

En la actualidad, estas pantallas se están volviendo muy populares ya que su ancho es la décima parte del ancho de un monitor convencional, y pesan solamente la sexta parte. [62] Además, al ser planas en lugar de levemente curvas, no tienen distorsión en las esquinas. Las pantallas planas se las pueden encontrar en diversos tamaños típicos de monitores de PC y también en tamaños más grandes, hasta de 60 pulgadas. [63]

1.2.6.3.2 Pantallas OLED

OLED son las siglas en inglés de diodo orgánico emisor de luz. Esta es una nueva tecnología de presentación que aprovecha las cualidades electro-luminarias de ciertos materiales orgánicos. [67]

Como se puede ver en la Figura 1-18, las pantallas OLED ofrecen un mayor brillo y colores saturados, en relación con otras tecnologías de pantalla plana como LCD, pero su precio es todavía demasiado alto. [67]

Figura 1-18 Comparación de una pantalla OLED con una LCD [67]

Figura 1-19 Pantalla OLED utilizada en una cámara digital Kodak [67]

En la actualidad esta tecnología está siendo utilizada para pantallas pequeñas, como las de cámaras digitales, pero se espera que en el 2004, algunos fabricantes como Samsung, Sanyo y Toshiba produzcan pantallas de hasta 17" con una resolución de 1280x768 píxeles. [67]

1.3 Librerías de Software para generación de gráficos por computador

Las librerías para generación de gráficos por computador son un tipo de *Application Software Interface* (API) utilizados para definir imágenes gráficas en dos y tres dimensiones. Los API son métodos específicos determinados por un sistema operativo o por una aplicación mediante los cuales un programador puede realizar requerimientos al sistema operativo o a otra aplicación. [63]

1.3.1 OpenGL

Open Graphics Library (OpenGL) es un estándar multiplataforma utilizado para la renderización y aceleración por hardware de imágenes en dos y tres dimensiones. La librería de software de OpenGL es compatible con todos los sistemas Windows, MacOS, Linux y Unix. [35]

Desde su introducción en 1992, OpenGL es el API más utilizado por la industria de la computación para la generación de gráficos por computador. Antes de OpenGL, la parte gráfica de cualquier aplicación debía ser reescrita para poder ser utilizada por distintos sistemas operativos. Con OpenGL, una aplicación puede crear los mismos efectos en cualquier sistema operativo, siempre y cuando tenga un adaptador compatible. [63]

OpenGL especifica una lista de comandos que realizan operaciones de dibujo o causan efectos especiales. Entre las capacidades de OpenGL están el modelado de transformaciones, mapeo de texturas, transparencias, efectos atmosféricos, entre otros. [63]

OpenGL es un lenguaje procesal, es decir que en lugar de describir como debe verse una escena, el programador detalla los pasos necesarios para lograr esa escena. [66]

OpenGL no incluye funciones para el manejo de ventanas o archivos. Cada sistema operativo tiene sus propias funciones y es responsable de implementar los medios necesarios para que OpenGL tenga el control de los mismos. [66]

Como se puede ver en la Figura 1-20, las aplicaciones de Windows utilizan la librería GDI32.DLL para comunicarse con la tarjeta de video. OpenGL en cambio, utiliza una interfaz 3D, especialmente diseñada para permitir la aceleración del hardware. [66]

Entre las principales ventajas de utilizar OpenGL están:

- Su portabilidad, lo cual permite a los programadores escribir aplicaciones que pueden correr en múltiples plataformas.
- Es un estándar abierto, es decir que cada compañía puede comprar una licencia y crear su propia implementación, lo cual permite actualizaciones constantes.
- Tiene muchas funciones y extensiones creadas por diversas compañías en áreas tales como juegos, militar, diseño asistido por computador, entre otras.
- Existe una extensa documentación en Internet, la cual incluye manuales de usuario y definiciones de clases. [36]

Entre las principales desventajas están:

- Existen demasiadas extensiones, por lo que el código se puede volver desordenado y confuso.
- A pesar de que la mayoría de tarjetas gráficas soportan OpenGL y DirectX, unas cuantas todavía tienen problemas con OpenGL.
- No es orientado a objetos. [36]

OpenGL puede hacer prácticamente lo mismo que DirectX pero en múltiples plataformas, por lo tanto si no se quiere tener la limitación de poder trabajar solamente en Windows, es recomendable utilizar OpenGL. [36]

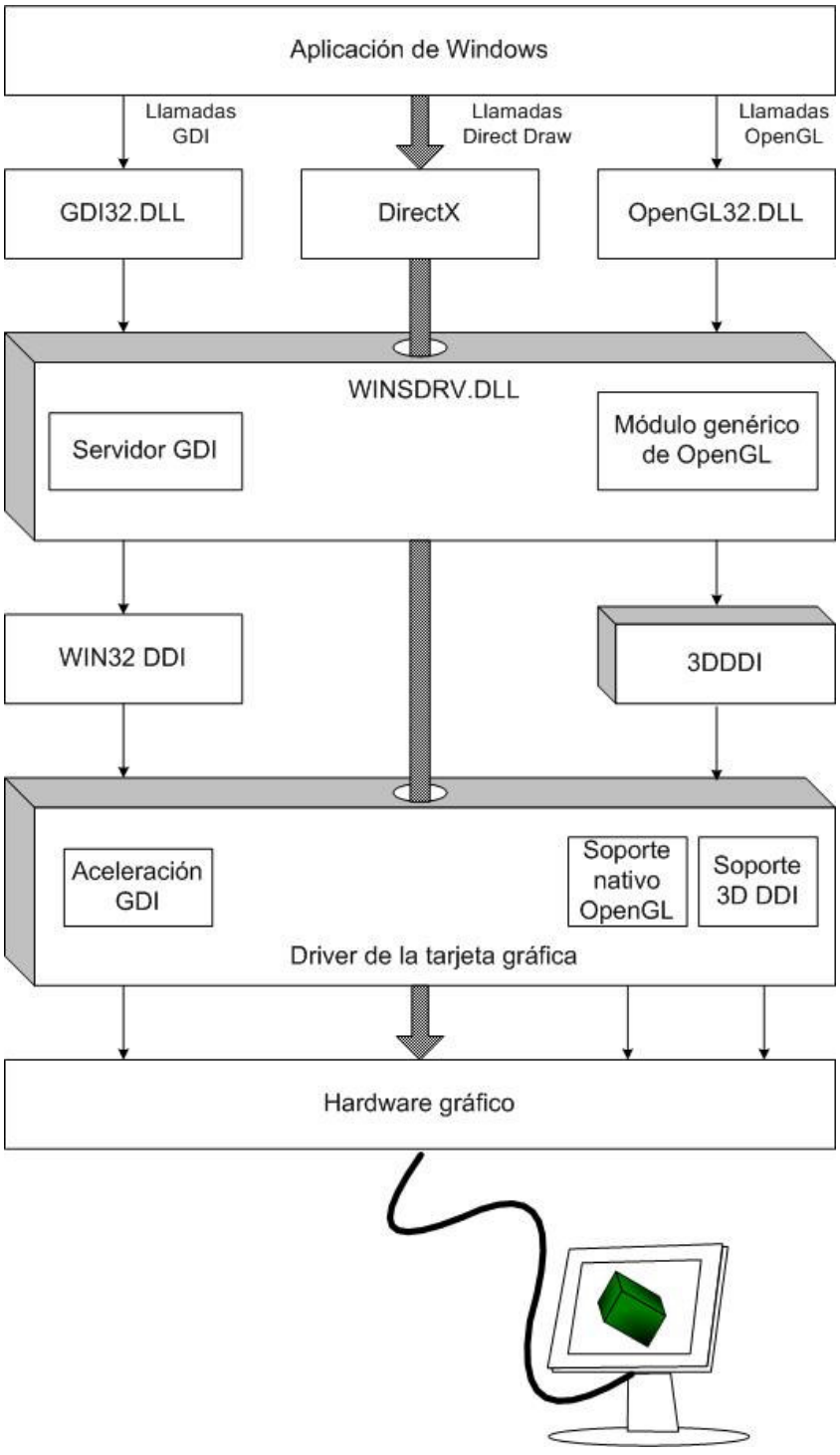


Figura 1-20 Diagrama de cómo funciona la aceleración gráfica en Windows [66]

1.3.2 DirectX

Microsoft DirectX es un API utilizado para crear y manejar imágenes gráficas y efectos multimedia en aplicaciones que corren en el sistema operativo Windows. [63]

Esta tecnología fue introducida en 1995, y es reconocida como un estándar para el desarrollo de aplicaciones en la plataforma Windows. [14]

DirectX está diseñado de tal forma que, si se tiene el *driver* adecuado, algunas funciones pueden ser realizadas por la tarjeta aceleradora de gráficos, liberando de trabajo al microprocesador. [63]

Como se puede ver en la Figura 1-20, DirectX está optimizado para un acceso directo al hardware gráfico, es decir evita las librerías gráficas de Windows y se comunica directamente con la tarjeta de video. [66]

Entre las principales ventajas de utilizar DirectX están:

- Permite una programación orientada a objetos.
- Al ser dependiente de la plataforma, no es necesario escribir código específico para cada proveedor. [36]

Entre las principales desventajas están:

- Solamente se puede utilizar en aplicaciones de Windows.
- Las actualizaciones solo se realizan una vez al año.
- Código extenso y muchas veces confuso.

- No es un estándar abierto. [36]

DirectX es una buena opción para programadores experimentados que solamente quieran utilizar aplicaciones de Windows. [36]

1.4 Procesamiento distribuido

1.4.1 Definición

En computación, el procesamiento en paralelo es a la división de las instrucciones de un programa entre múltiples procesadores con el objetivo de que la ejecución del programa demore menos tiempo. [63]

El procesamiento distribuido, en cambio, supone no solamente la distribución de tareas entre diferentes procesadores, sino entre diferentes computadores, por lo que la comunicación se realiza a través de una red de área local. [37]

1.4.2 Beneficios

El principal beneficio de utilizar sistemas paralelos o distribuidos es realizar tareas más rápidamente. [24]

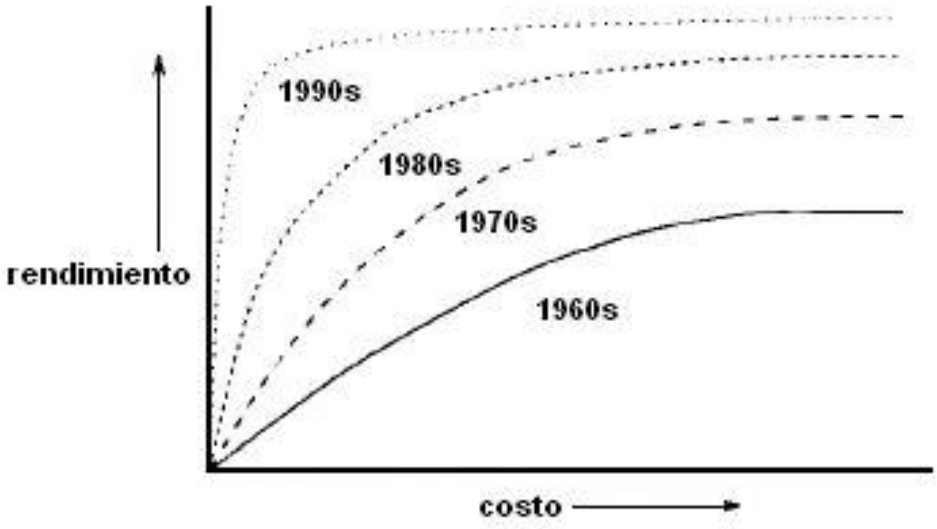


Figura 1-21 Evolución del rendimiento y el costo de los computadores [24]

Como se puede ver en la Figura 1-21, el poder de procesamiento secuencial es limitado, por lo tanto se debe utilizar el procesamiento en paralelo para pasar este límite. Además la utilización de sistemas paralelos y distribuidos ha permitido que sistemas con computadores de bajo costo alcancen niveles de rendimiento similares a computadores mucho más caros. [24]

Al conseguir una ejecución más rápida, se obtienen otros beneficios, tales como la realización de múltiples tareas en un periodo de tiempo determinado; y, disminuir el tiempo necesario para completar una tarea. Además, se pueden resolver problemas más grandes y complicados, que probablemente no hubieran podido ser analizados por un solo computador. [1]

1.4.3 Tipos de sistemas distribuidos

Los sistemas distribuidos se clasifican de acuerdo a cómo procesan los flujos de instrucciones y de datos. Los principales tipos de sistemas son: SISD, SIMD, MISD, MIMD.

1.4.3.1 SISD

SISD (*Single Instruction Single Data*), es en realidad un sistema con un solo procesador en el cual existe un flujo único de instrucciones y de datos, como se puede ver en la Figura 1-22. [24]

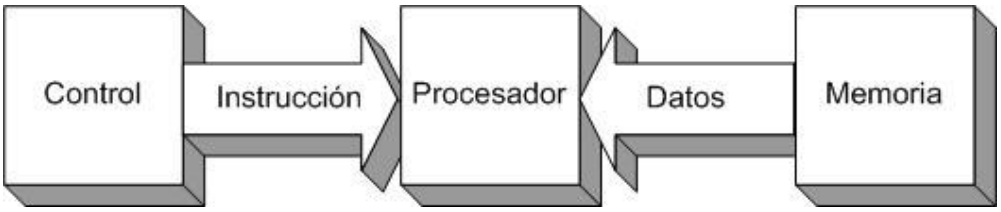


Figura 1-22 Modelo SISD [29]

En la actualidad algunos servidores poseen más de un procesador, pero cada uno ejecuta instrucciones que no están relacionadas entre sí, por lo tanto estos sistemas se los considera también máquinas SISD. Ejemplos de este tipo de máquinas son la mayoría de computadores o estaciones de trabajo. [16]

1.4.3.2 SIMD

SIMD (*Single Instruction Multiple Data*), es un sistema en el cual la misma instrucción es ejecutada de manera sincrónica por todos los procesadores, como se puede ver en la Figura 1-23. [24]

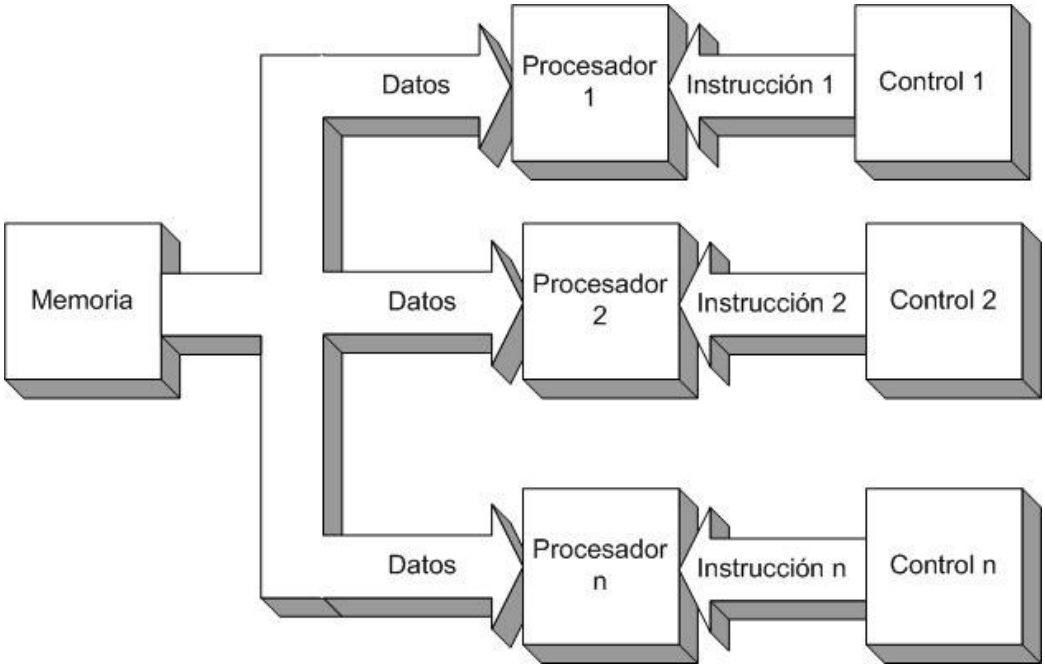


Figura 1-23 Modelo SIMD [29]

Un ejemplo de este tipo de sistemas es el SETI@home, que es un experimento científico que utiliza computadores conectados a Internet para analizar datos de telescopios en la búsqueda de inteligencia extraterrestre. [50]

1.4.3.3 MISD

MISD (*Multiple Instruction Single Data*), es un sistema en el cual se segmenta el programa y se lo divide entre múltiples procesadores, de tal manera que cada uno realiza una instrucción diferente, como se puede ver en la Figura 1-24. [24]

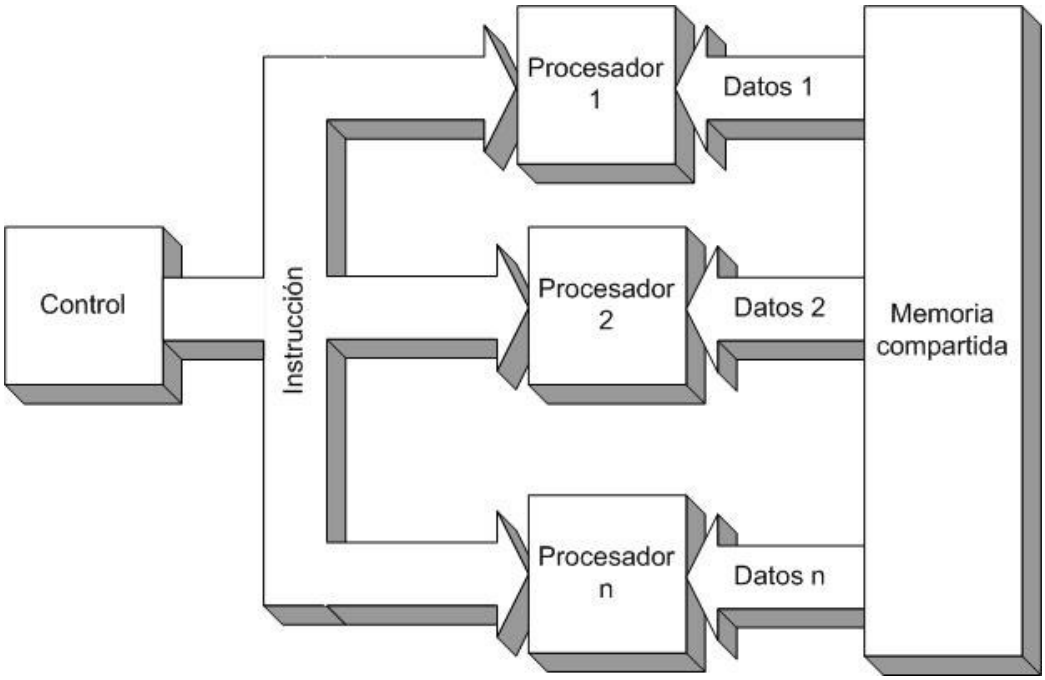


Figura 1-24 Modelo MISD [29]

Este tipo de máquinas son solamente teóricas, ya que no se ha construido ningún sistema práctico de esta clase. [16]

1.4.3.4 MIMD

MIMD (*Multiple Instruction Multiple Data*), es un sistema en el cual se ejecuta un programa diferente en cada procesador, como se puede ver en la Figura 1-25. [24]

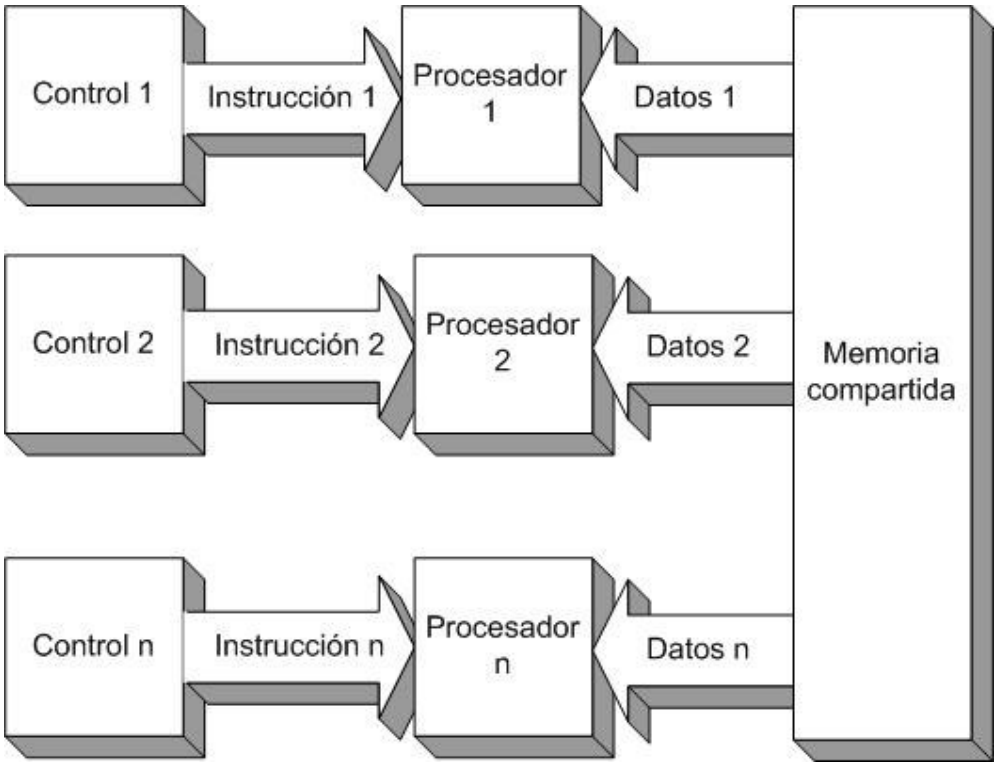


Figura 1-25 Modelo MIMD [29]

La diferencia con los sistemas SISD de múltiples procesadores es que las instrucciones y los datos están relacionados, ya que representan diferentes partes de una misma tarea. Existen gran variedad de

sistemas MIMD, tales como el nCUBE 3, el Cray Y-MP T94, entre otros. [16]

1.4.4 Pasos para ejecutar un programa en paralelo

Para ejecutar un programa en paralelo se deben seguir los siguientes pasos:

- *División*: Consiste en dividir el trabajo entre los múltiples procesadores.
- *Comunicación*: Consiste en enviar las instrucciones a cada procesador y recibir los resultados calculados por los mismos.
- *Sincronización*: Consiste armonizar y combinar los resultados enviados por cada procesador.

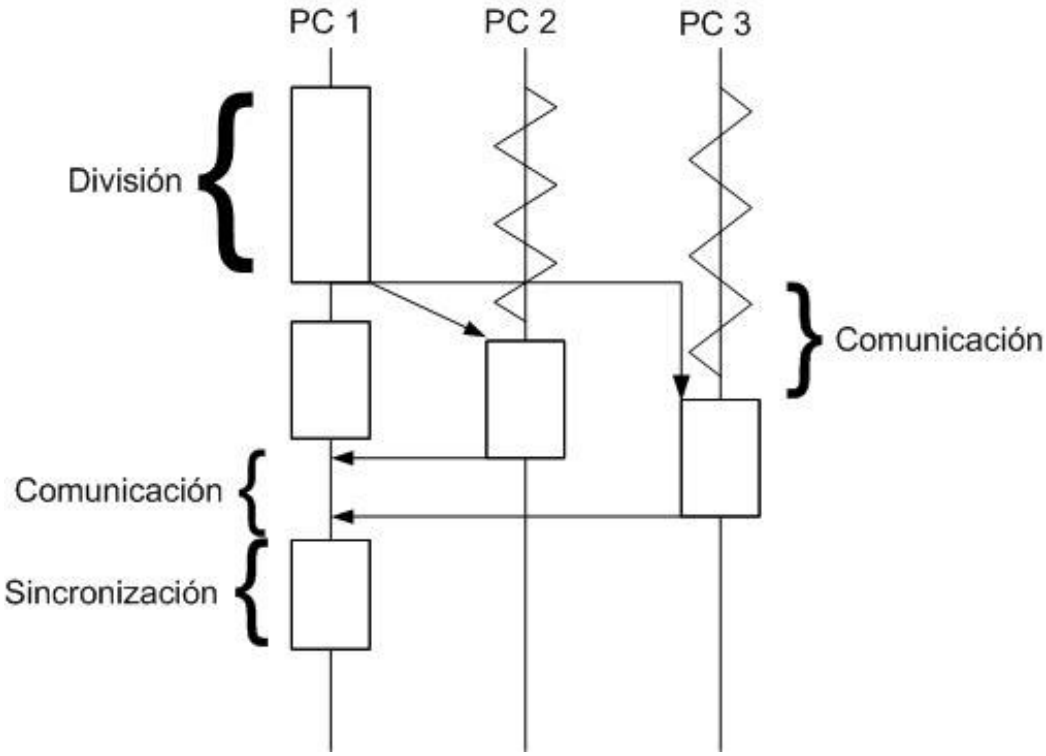


Figura 1-26 Pasos para ejecutar un programa en paralelo en paralelo [24]

1.4.5 Rendimiento de un sistema distribuido

1.4.5.1 Tiempo de ejecución

El tiempo de ejecución de un programa en paralelo es igual al tiempo de procesamiento más el tiempo de comunicación entre los procesadores y el tiempo de bloqueo. [24]

El tiempo de comunicación, como su nombre lo dice, es el tiempo que se demoran los computadores en pasarse la información entre sí. [24]

El tiempo de bloqueo se refiere al tiempo que un procesador espera a que otro termine su tarea para continuar con la siguiente instrucción. [24]

1.4.5.2 Aceleración o *Speed-up*

La aceleración o *speed-up* es la relación que existe entre el tiempo necesario para resolver un problema de manera secuencial y el tiempo utilizado para resolver el mismo problema en paralelo. [24]

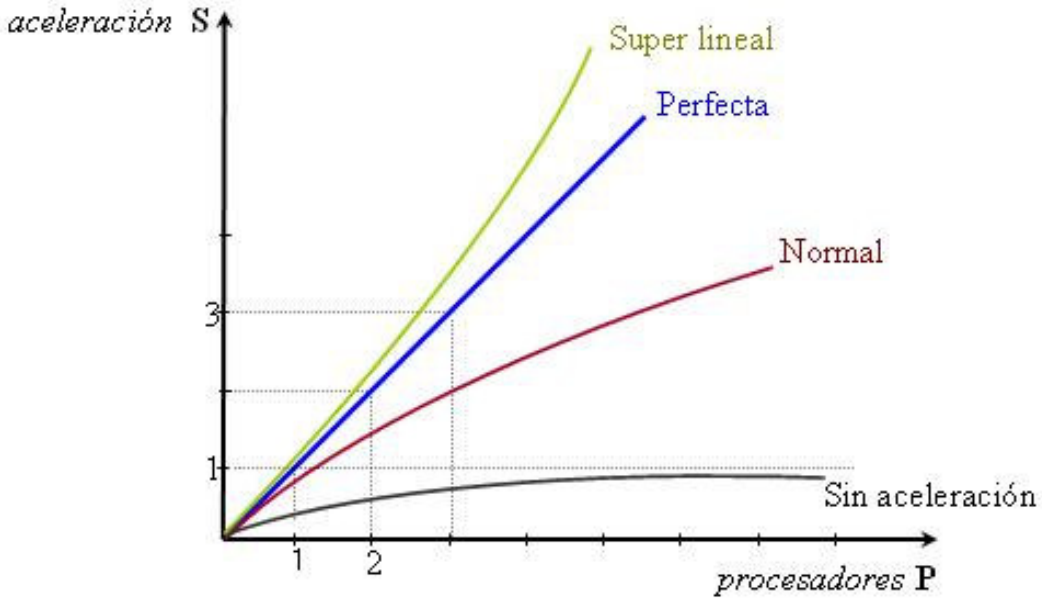


Figura 1-27 Aceleración de sistemas paralelos [24]

Como podemos ver en la Figura 1-27, se pueden dar los siguientes casos:

- S>P: Aceleración súper lineal (en realidad no existe).
- S=P: Sistema paralelo perfecto.
- 1<S<P: Utilizar el sistema paralelo es más rápido que el secuencial. Es el caso normal.
- S<=1: El procesamiento en paralelo es tan o más lento que el secuencial. [24]

1.4.5.3 Eficiencia

La eficiencia de un sistema distribuido es la relación entre la aceleración o *speed-up* del sistema y el número de procesadores que se utilizan. En un sistema perfecto, la eficiencia sería igual a uno. [24] En la práctica se ha demostrado que mientras más procesadores se utilicen la eficiencia es menor. Esto se debe a que se utiliza más tiempo en la división, comunicación y sincronización de los datos. [24]

CAPÍTULO 2

2 ANÁLISIS

En este capítulo se analizan las posibles soluciones de hardware, las cuales son el uso de supercomputadores, de computadores de alto rendimiento o de un *cluster* computadores. También se analizan las distintas soluciones de software para crear un *cluster* (*Beowulf*, *Oscar*, *CLIC*). Además, se investigan las distintas soluciones de software para el servidor X (*DMX*, *VNCWall*), para paralelización de instrucciones gráficas (*WireGL*, *Chromium*) y para la visualización de datos (*OpenDX*, *Vis5d+*). Por último, se describe la solución escogida y su alcance.

2.1 Soluciones de Hardware

El sistema de visualización necesita un alto poder de procesamiento para poder realizar los cálculos necesarios para mostrar simulaciones y animaciones. Para lograr este poder de procesamiento, tenemos las siguientes opciones: un supercomputador, un computador de alto rendimiento, o un *cluster* de computadores.

2.1.1 Supercomputador

Supercomputador es un término utilizado para describir a los computadores más rápidos disponibles en la actualidad. Estos computadores generalmente son utilizados para procesar gran cantidad de datos, por ejemplo en simulaciones científicas, animaciones gráficas, análisis estructurales, diseño electrónico, meteorología, etc. El fabricante de supercomputadores más conocido es *Cray Inc*. [8]

Debido a que los supercomputadores realizan principalmente operaciones con números, su velocidad es medida en operaciones de punto flotante por segundo o FLOPS, por sus siglas en inglés. Para lograr una alta velocidad, los componentes de los supercomputadores se juntan para minimizar la distancia que tienen que recorrer las señales electrónicas. Los supercomputadores también utilizan

técnicas especiales para evitar el calor en los circuitos y prevenir que se quemen debido a su proximidad. [8]

Los supercomputadores tienen un costo muy elevado. Por ejemplo, la *ASCI Option Red* que posee 9,000 procesadores *Pentium Pro*, y cuya velocidad llega hasta 1.8 teraflops, costó 55 millones de dólares. La *ASCI Option Blue-Pacific*, que posee 4,096 procesadores diseñados y construidos exclusivamente para ese supercomputador por IBM, y cuya velocidad llega hasta los 3.2 teraflops, costó 93 millones de dólares. [8]

Si revisamos entre los supercomputadores más "asequibles," encontramos que un *SGI Origin* con 64 procesadores y que alcanza los 24 Gigaflops cuesta aproximadamente 1.8 millones de dólares. [37]

2.1.2 Computadores de alto rendimiento

Los computadores de alto rendimiento son máquinas que pueden resolver problemas grandes en un periodo de tiempo razonable. Esto lo logra haciendo las operaciones secuenciales más rápidamente y realizando procesos en paralelo. [39]

Los computadores de alto rendimiento están caracterizadas por:

- Cálculos rápidos.
- Memoria grande.
- Interconexión de alta velocidad.

- Entradas y salidas de alta velocidad. [39]

Algunos constructores de computadores de alto rendimiento son *Silicon Graphics*, *Hewlett-Packard*, *IBM*, *Tandem*, entre otros. [39]

Los computadores de alto rendimiento son utilizados para múltiples aplicaciones, entre las principales están la predicción del clima, análisis del genoma humano, pruebas de choques de carros, diseño de aviones, inteligencia artificial, exploración sísmica, entre otras. [39]

Silicon Graphics es el principal constructor de computadores de alto rendimiento. Entre sus productos se encuentran la serie *Onyx* y la *Octane*. [52]

Figura 2-1 *Silicon Graphics Octane* [10]

El *Silicon Graphics Octane* (Figura 2-1) es una estación de trabajo de un gran rendimiento gráfico. Este computador optimiza las visualizaciones, clarifica problemas y acelera el proceso de encontrar soluciones. Puede tener uno o dos procesadores, y su precio empieza en cuarenta mil dólares. [10] En pruebas, el *Octane* ha alcanzado los 148 megaflops. [6]

Figura 2-2 *Silicon Graphics Onyx2* [52][10]

El *Silicon Graphics Onyx* (Figura 2-2) es un computador con la capacidad de utilizar de 2 a 32 tarjetas gráficas en paralelo, de tal

forma que actúe como una sola tarjeta gráfica grande y poderosa. También tiene la capacidad de trabajar hasta con 64 procesadores. La versión más económica de esta estación de trabajo (con dos tarjetas gráficas y dos procesadores) cuesta 45,000 dólares. Mientras que la versión más completa (con 32 tarjetas gráficas y 64 procesadores) está alrededor de los 1.2 millones de dólares. [30]

2.1.3 *Cluster* de computadores

Un *cluster* es un grupo de computadores interconectados entre sí que forman un sistema potente. Este sistema se muestra al usuario como un solo computador con gran capacidad. [63]

Los *clusters* están formados por una red de computadores ordinarios, cada uno de los cuales tiene instalado el sistema operativo (generalmente Linux) y un software que le permite participar en el procesamiento en paralelo. [3]

Los *clusters* de computadores pueden ser utilizados como una alternativa, de un costo relativamente bajo, para procesamiento en paralelo de aplicaciones científicas. [63] Por ejemplo el *Avalon*, un *cluster* de 140 computadores que está entre las 100 máquinas más potentes del planeta, y que alcanza una velocidad de 47.7 gigaflops, costó solamente 340,000 dólares. [37]

Además del costo mucho menor, otra ventaja importante de los *clusters* es que si ocurre algún daño en una máquina, es mucho más sencillo de resolver. Por ejemplo si se daña el procesador de un nodo, es mucho más fácil sustituir ese procesador que encontrar los repuestos de un supercomputador. [3]

Las principales desventajas de los *clusters* es que no hay mucho software capaz de tratar al *cluster* como un único sistema. Además, las redes de computadores no están diseñadas para el procesamiento en paralelo, por lo cual muchas veces el ancho de banda es demasiado bajo y su latencia demasiado alta en comparación con los buses de datos que utilizan los supercomputadores. [3]

Esta última desventaja puede ser superada mediante el uso de otras alternativas al Ethernet, tales como InfiniBand.

InfiniBand es una arquitectura que permite la conexión de redes a alta velocidad. Mientras que Ethernet llega solamente a 1 gigabit por segundo, la versión 4x de InfiniBand llega a 10 gigabits por segundo, y en la actualidad se está trabajando en una versión 12x, que triplicaría esa velocidad. El problema con esta tecnología es que todavía es cara en comparación con Ethernet. Un juego con un *switch* y tarjetas para conectar cuatro computadores está aproximadamente en diez mil dólares. [51]

2.2 Métodos de construcción de *clusters*

Existen múltiples opciones para construir *clusters* de computadores. En esta sección se van a analizar tres: *Beowulf* (la más antigua), *OSCAR* (una de las más comunes) y *CLIC*.

Cabe recalcar que todas estas opciones son sobre Linux, ya que proveen software abierto el cual puede ser utilizado sin costo alguno.

2.2.1 Beowulf

Beowulf es un método para construir un *cluster* de computadores utilizando computadores personales interconectados mediante una red de área local, el cual corre programas escritos para procesamiento en paralelo. [63]

El Beowulf original fue desarrollado en 1994 por el Centro de Excelencia en Datos del Espacio y Ciencias de la Información (CESDIS), que pertenece a la NASA. Este *cluster* consistía en 16 computadores Intel DX4 conectadas a través de una red Ethernet de 10 Mbps. [63]

Un *cluster* Beowulf utiliza una arquitectura de multicomputadores, como se puede ver en la Figura 2-3. Generalmente tiene un solo nodo maestro y varios nodos de cómputo o de cálculo, interconectados por una red. [19]

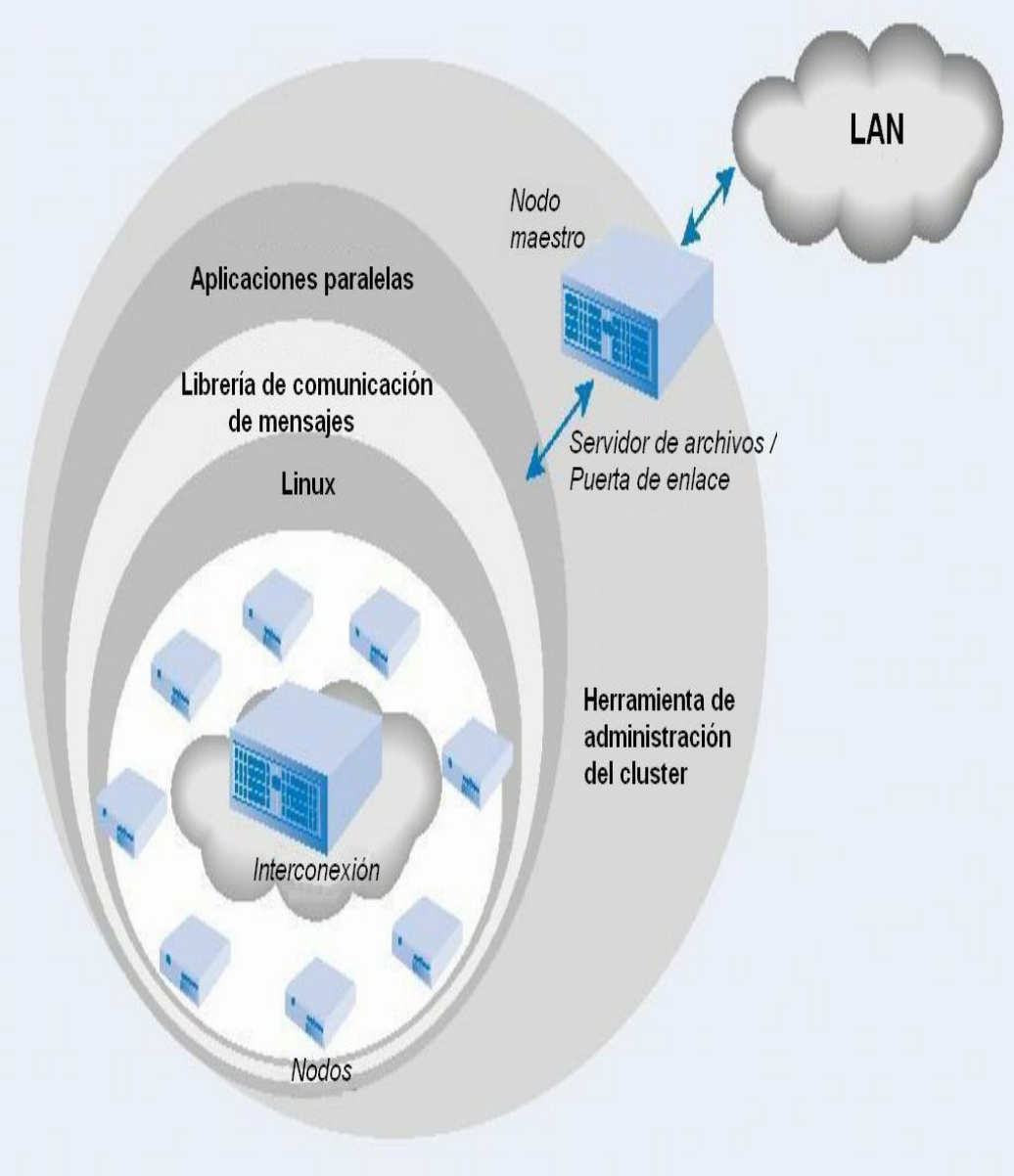


Figura 2-3 Vista lógica de un *cluster* Beowulf [19]

El nodo maestro actúa como servidor para el sistema de archivos y además es la puerta de enlace con el mundo exterior. Los nodos de cálculo tienen como única tarea ejecutar trabajos en paralelo. Por esta

razón en la mayoría de los casos, estos nodos no tienen tarjetas de video, monitores, teclados ni ratones. El acceso a estos nodos se da exclusivamente a través del nodo maestro. [19]

Desde la perspectiva del usuario, el *cluster* Beowulf es un solo computador al cual se envía un programa escrito con instrucciones en paralelo. El nodo maestro es el encargado de distribuir las instrucciones entre los demás nodos. [19]

Vale recalcar que aunque la mayoría de los *cluster* Beowulf han utilizado Linux como sistema operativo, también existen *clusters* que utilizan Windows NT o Windows 2000. [19]

A pesar de que Beowulf no es en sí mismo un paquete de software, existen paquetes comerciales que incluyen todo lo necesario para construir un *cluster* Beowulf. Estos paquetes están basados en RedHat e incluyen el kernel, los utilitarios de Linux, la interfaz gráfica, entre otros programas. [7]

2.2.2 OSCAR

Open Source Cluster Application Resources (OSCAR) es un proyecto desarrollado por el Grupo *Open Cluster*, que consiste en un paquete de software que contiene todos los programas necesarios para instalar, construir, mantener y utilizar un *cluster* sobre Linux de tamaño

modesto. Este paquete fue desarrollado para facilitar la construcción de un *cluster*. [38]

A pesar de la existencia de material de referencia para la construcción de *clusters*, esta tarea resultaba muy tediosa debido a que se debían adquirir, compilar y probar una gran cantidad de componentes de hardware y software. Por esta razón se creó OSCAR en abril del 2000. Su objetivo principal es que la instalación, configuración y administración de un *cluster* de tamaño modesto sea relativamente fácil. [12]

La arquitectura de un *cluster* OSCAR es muy similar a la de de un *cluster* Beowulf. Se configura un nodo como maestro, el cual provee los servicios a los nodos de cómputo. En el *cluster* Beowulf, una vez que se ha configurado el nodo maestro, se deben construir cada uno de los nodos de cálculo. OSCAR, en cambio, asiste en la configuración del nodo principal y construye los nodos de cálculo basados en una descripción especificada por el usuario. Esta construcción y configuración reduce considerablemente el esfuerzo y la pericia necesaria para conformar un *cluster*. [12]

OSCAR posee la infraestructura principal, herramientas de administración y configuración, herramientas y servicios para computación de alto rendimiento (como monitoreo de carga,

repartición de instrucciones, entre otros), y herramientas de seguridad. [12]

2.2.3 CLIC

Cluster LInux pour le Calcul (CLIC) es un proyecto desarrollado por la Agencia Francesa de Nuevas Tecnologías (RNTL) cuyo objetivo es el desarrollo de un software de distribución libre que sirva para construir un *cluster* de computadores. Este software debe cubrir las necesidades de despliegue, administración y programación de *clusters* precisas para cálculos masivos e intensivos. [26]

Al igual que OSCAR, lo que hace CLIC es facilitar la construcción de un *cluster* de tipo Beowulf, juntando todos los componentes necesarios en un solo paquete que debe ser instalado en el servidor y en los nodos de cálculo.

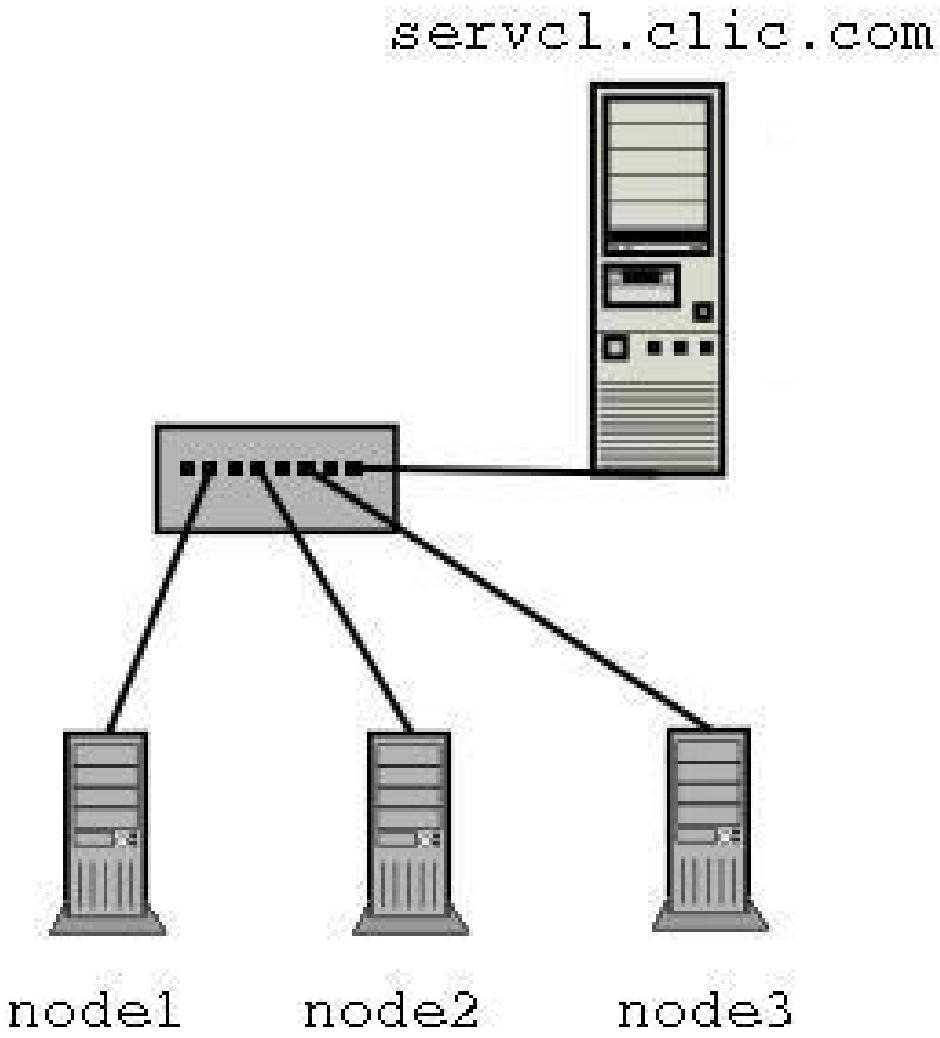


Figura 2-4 Arquitectura de un *cluster* CLIC [26]

La Figura 2-4 muestra la arquitectura de CLIC, que consiste en un computador que actúa como servidor y se comunica con los nodos de cálculo a través de una red de área local.

El procedimiento de instalación de CLIC consiste en instalar y configurar el servidor, luego se instala uno de los nodos de cómputo y se replica esta instalación en los demás nodos. [26]

CLIC provee herramientas de administración y auto configuración, herramientas de monitoreo, herramientas de seguridad, un gran número de librerías matemáticas, e inclusive NetJuggler, un programa que permite la paralelización de imágenes 3D y realidad virtual. [26]

Una de las principales ventajas que posee CLIC sobre OSCAR es que posee herramientas de auto configuración, con lo cual resulta mucho más sencillo construir el *cluster*. Además, al ser CLIC un software desarrollado para realizar cálculos intensivos, posee un mayor número de librerías matemáticas las cuales pueden resultar útiles en el momento de realizar cómputos.

2.3 Software para el entorno gráfico

En el sistema operativo Linux, el entorno o interfaz gráfica es provista por el servidor X. El sistema X Windows versión 11, mejor conocido como X11, es el sistema gráfico estándar para Unix. [48]

El X11 permite a las aplicaciones dibujar píxeles, líneas, texto e imágenes en la pantalla. También posee librerías adicionales que permiten dibujar fácilmente interfaces con el usuario tales como botones, campos de texto, entre otras. [48]

Los ambientes de escritorio de Linux, tales como KDE y GNOME, corren sobre el X11. [48]

En esta sección se van a analizar dos alternativas para la distribución del entorno gráfico en varias pantallas. La primera es DMX y la segunda es *VNCWall*.

2.3.1 DMX

Distributed Multihead X (DMX) es un servidor X *front-end* que controla la interacción con el usuario y que es capaz de controlar múltiples servidores *back-end* los cuales pueden formar una pantalla de cualquier tamaño. Estos servidores X pueden correr en la misma máquina o en otros computadores. [15] Esto último es muy importante ya que antes de la introducción de este programa, la distribución de una pantalla estaba limitada a la cantidad de ranuras AGP o PCI que poseía un computador.

La estructura general de DMX es la siguiente: Un servidor X *front-end* actúa como *proxy* de una serie de servidores *back-end*, los cuales manejan toda la renderización de imágenes. La relación entre el *proxy* y los servidores es transparente para el usuario. Para él, el computador que actúa de *proxy* controla una gran pantalla, sin importar como divide las instrucciones. [27]

DMX puede ser dividido en dos partes principales: dispositivos de entrada y llamadas de renderización. Los dispositivos de entrada básicos, tales como teclado y ratón deben estar conectados directamente al *front-end*. Otros dispositivos son procesados por la extensión XInput, y pueden ser manejados por el servidor *proxy* o por cualquiera de los otros computadores. [27]

Las llamadas de renderización son receptadas por el servidor *proxy*, sin embargo, son divididas y enviadas al computador *back-end* correspondiente mediante llamadas con la librería X11.

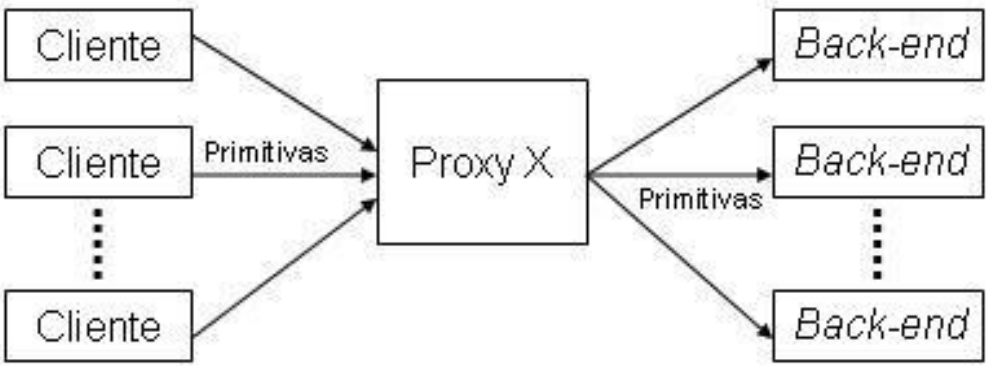


Figura 2-5 Diagrama de funcionamiento de DMX [61]

Como se puede ver en la Figura 2-5, el computador que actúa como *proxy* acepta los comandos de visualización, y los distribuye entre los demás nodos. Vale recalcar que para el envío de la información de

píxeles, DMX utiliza primitivas de dos dimensiones (es decir polígonos, tales como cuadrados, pentágonos, entre otros) para reducir el ancho de banda necesario para pasar los mensajes.

2.3.2 VNCWall

Virtual Network Computing es un software de control remoto que permite al usuario interactuar con un computador (el "servidor") usando un programa simple (el "visualizador" o "espectador") en otro computador. [44]

VNCWall es un programa utilitario que permite mostrar una o más sesiones de VNC en una pantalla de visualización. Con este programa una sesión VNC puede ser mostrada en múltiples proyectores de una pantalla. VNC permite especificar el número de proyectores que conforman la pantalla y además permite determinar cuales mostrarán la sesión VNC. [59]

El servidor VNC fue modificado para que VNCWall pueda manejar pedidos de múltiples secciones rectangulares de una pantalla. Esto permite que cada nodo se conecte al servidor y pida una sección diferente del escritorio, con lo cual se crea una pantalla de visualización. [61]

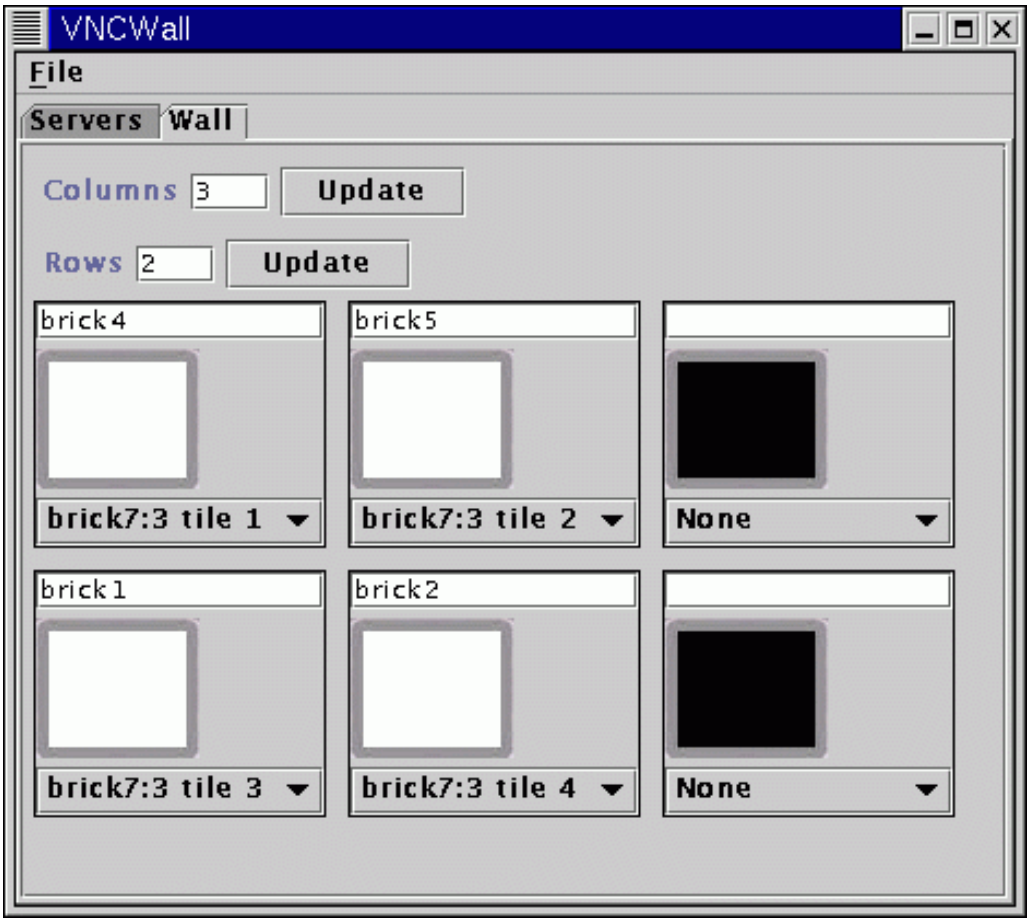


Figura 2-6 Programa de configuración de VNCWall [59]

VNCWall posee una interfaz con el usuario que facilita su administración. En la Figura 2-6 se puede ver la configuración de una pantalla de tres columnas y dos filas.

La principal desventaja de VNCWall es que se necesita que una de los computadores actúe exclusivamente como servidor. En cambio en DMX el computador que actúa como servidor también puede ser un

back-end, con lo cual se necesita un computador menos para el sistema.

Otra desventaja, es que VNCWall no permite la distribución del procesamiento gráfico, el servidor hace todos los cálculos y manda a dibujar los píxeles correspondientes a cada pantalla. En cambio, DMX si permite esta distribución, y además no manda a dibujar píxeles sino primitivas, por lo tanto, DMX es más rápido que VNCWall.

2.4 Software para la distribución en paralelo de instrucciones gráficas

Con el software para el entorno gráfico se consiguió dividir el escritorio de tal manera que se podía determinar que pantalla iba a dibujar cada parte, pero si todo el trabajo se lo cargamos al computador que hace de servidor, este proceso va a ser muy lento. Por esta razón es necesario utilizar un software que permita la distribución de las instrucciones gráficas, de tal manera que cada uno de los computadores que conforman el *cluster* realice una parte del procesamiento gráfico.

En esta sección se van a analizar dos paquetes de software que realizan la distribución en paralelo de instrucciones gráficas: *WireGL* y *Chromium*.

2.4.1 WireGL

WireGL es un proyecto desarrollado por el Laboratorio de Gráficos por Computador de la Universidad de Stanford que explora el uso de sistemas de renderización basados en *clusters* de computadores. [64]

El paquete de software WireGL permite la unificación del poder de renderización de las tarjetas aceleradoras de gráficos de los nodos de un *cluster*. Para lograr esto, WireGL provee una interfaz virtual al hardware gráfico a través de OpenGL. [21]

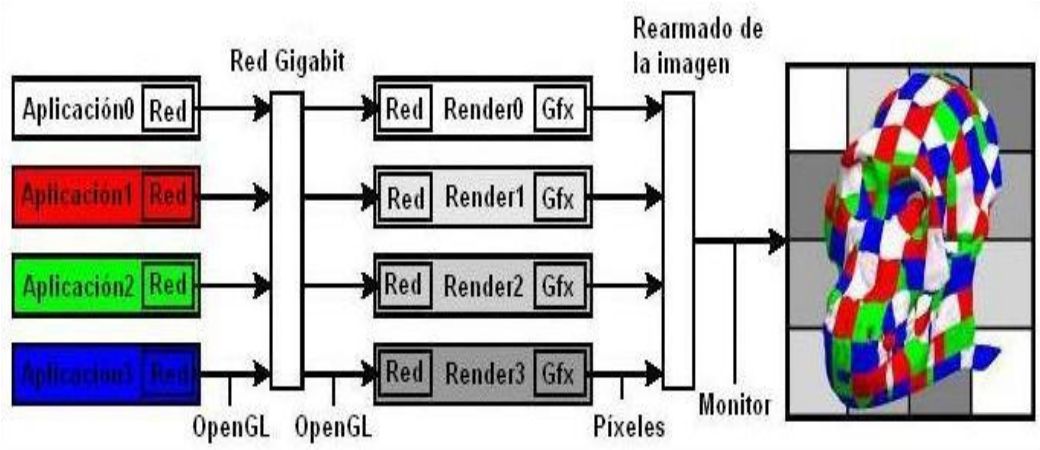


Figura 2-7 Diagrama de los componentes de WireGL [21]

Como se puede ver en la Figura 2-7, WireGL está compuesto por nodos de aplicación o clientes, nodos de renderización o servidores y un dispositivo de salida, el cual puede ser una pantalla de visualización o simplemente un monitor. [21]

WireGL no pone restricciones en el número de clientes o servidores. El software tiene la capacidad de trabajar en ambientes heterogéneos, en los cuales los clientes y servidores pueden estar corriendo diferentes sistemas operativos. La librería cliente de WireGL está implementada como un reemplazo de la librería OpenGL en Windows, Linux o IRIX. [21]

WireGL está implementado usando el modelo cliente/servidor. El cliente actúa como la fuente de los gráficos, generando comandos gráficos que deben ser creados por el *cluster*. Los servidores reciben los comandos gráficos de la red y los renderizan. [64]

En la Figura 2-8, se muestra el diagrama de funcionamiento de WireGL en un sistema con un cliente y cuatro servidores. El cliente (en la parte izquierda del diagrama) corre la aplicación, y reemplaza la librería de OpenGL (opengl32.dll para Windows o libgl.so para Linux) con la librería propia de WireGL, la cual exporta la misma interfaz que OpenGL. Esto permite que la aplicación no deba ser modificada para trabajar con WireGL. El cliente transmite los comandos gráficos a los servidores. Estos son responsables de la renderización de estos comandos. En este diagrama, cada servidor está conectado a un dispositivo de salida (el cual puede ser un proyector), por lo tanto se los configura para que muestren solamente la porción de la pantalla que les corresponde. [64]

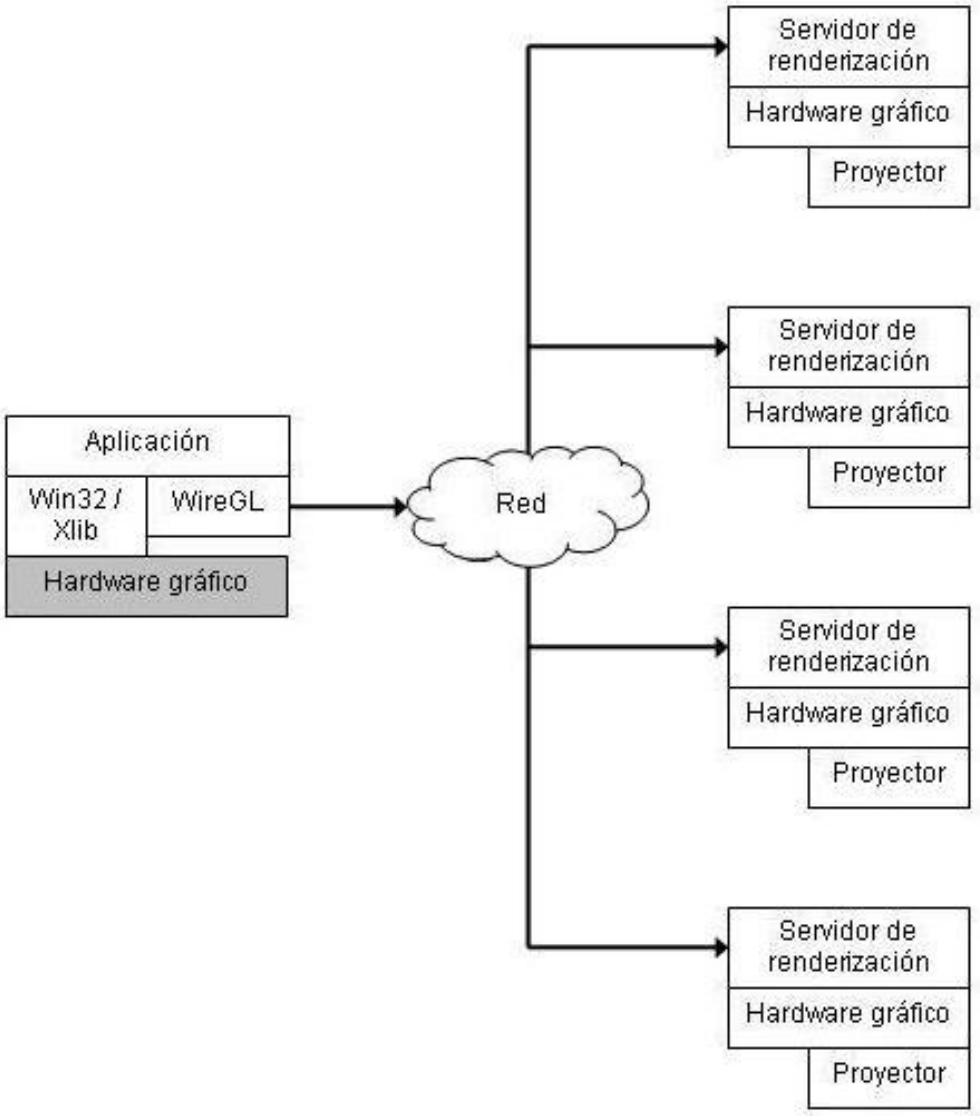


Figura 2-8 Diagrama de funcionamiento de WireGL [64]

WireGL ha sido probado por la Universidad de California y no funciona con la mayoría de programas de visualización actuales, tales como

Performer, OpenDX, y vis5d. Sin embargo si funciona con VMD. [53] Esto se debe a que WireGL no tiene implementados varios comandos de OpenGL que son necesarios para correr esos programas.

2.4.2 Chromium

Chromium es un sistema que permite la renderización interactiva en *clusters* de computadores. Chromium fue desarrollado por la Universidad de Stanford y está derivado del proyecto WireGL. [9]

Chromium es un software basado en WireGL, pero mejorado. WireGL requiere que todos los objetos geométricos sean transmitidos a través de la red, pero las redes no son lo suficientemente rápidas para mantener ocupadas a las tarjetas gráficas remotas. En cambio Chromium no requiere que todos los gráficos sean renderizados en un nodo diferente al que fueron originados, por lo que se puede modificar un dibujo sin que haya tráfico por la red. [20]

Chromium puede correr en múltiples sistemas, tales como Linux, IRIX, Windows, etc. [9]

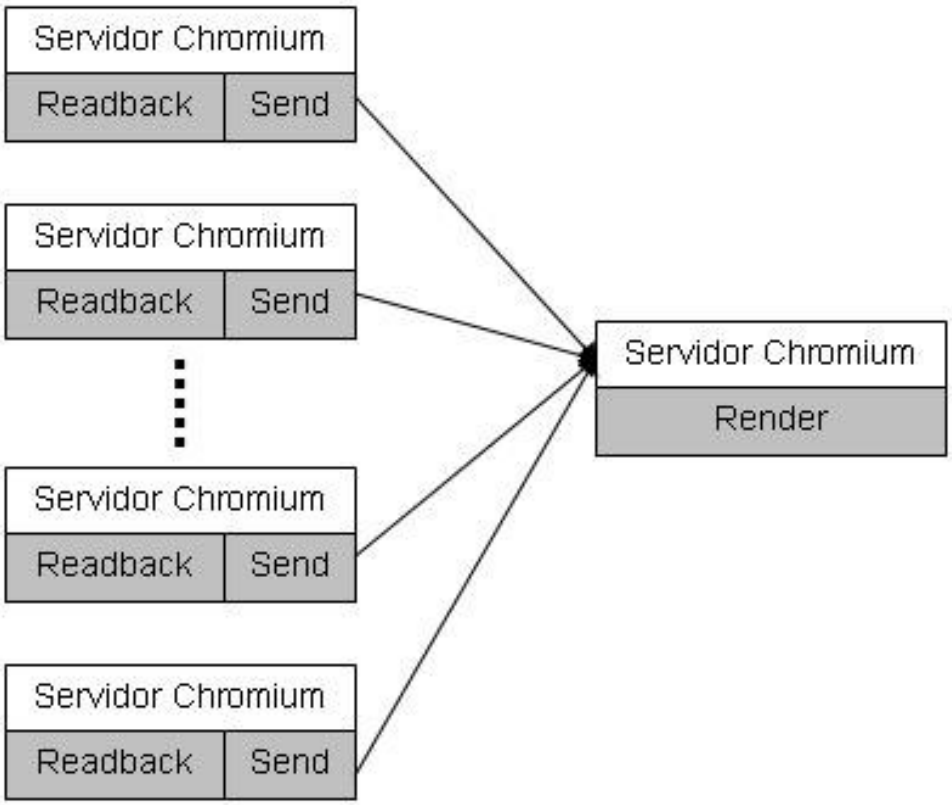


Figura 2-9 Configuración de Chromium con un solo dispositivo de salida [20]

En la Figura 2-9, se muestra una posible configuración de Chromium. En esta, los nodos de una aplicación paralela renderizan su parte del gráfico en el hardware local. Luego se transmite esa información a un servidor, el cual combina los resultados para producir la imagen final.

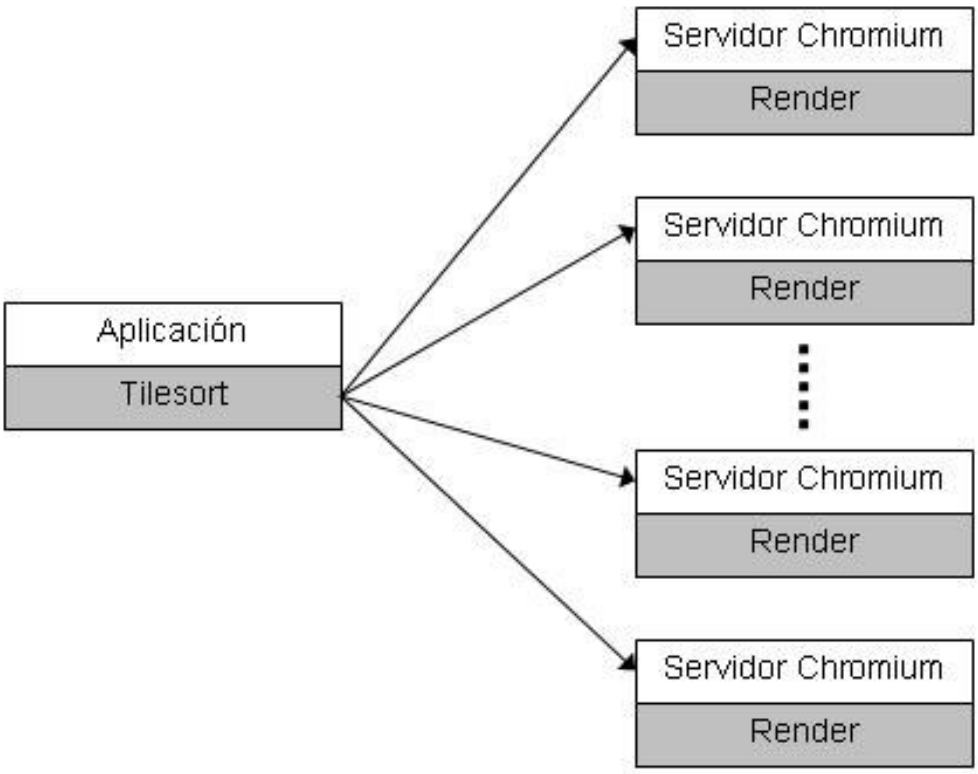


Figura 2-10 Configuración de Chromium múltiples servidores [20]

En la Figura 2-10, en cambio se tiene una máquina corriendo la aplicación y múltiples servidores encargados de renderizar las imágenes. Esta configuración es la típica de una pantalla de visualización.

Chromium ha sido probado por la Universidad de California y funciona con distintos programas de visualización, tales como Performer, OpenDX, VMD, Vis5d, entre otros. [53]

La principal ventaja de Chromium sobre WireGL es que se han implementado muchas más librerías de OpenGL en Chromium, razón por la cual existen gran cantidad de programas que si corren en Chromium pero que no corren en WireGL.

Otra desventaja de WireGL es que es un proyecto que no se sigue desarrollando, es más, Chromium es la continuación de WireGL, por lo que es una versión más nueva y mejorada.

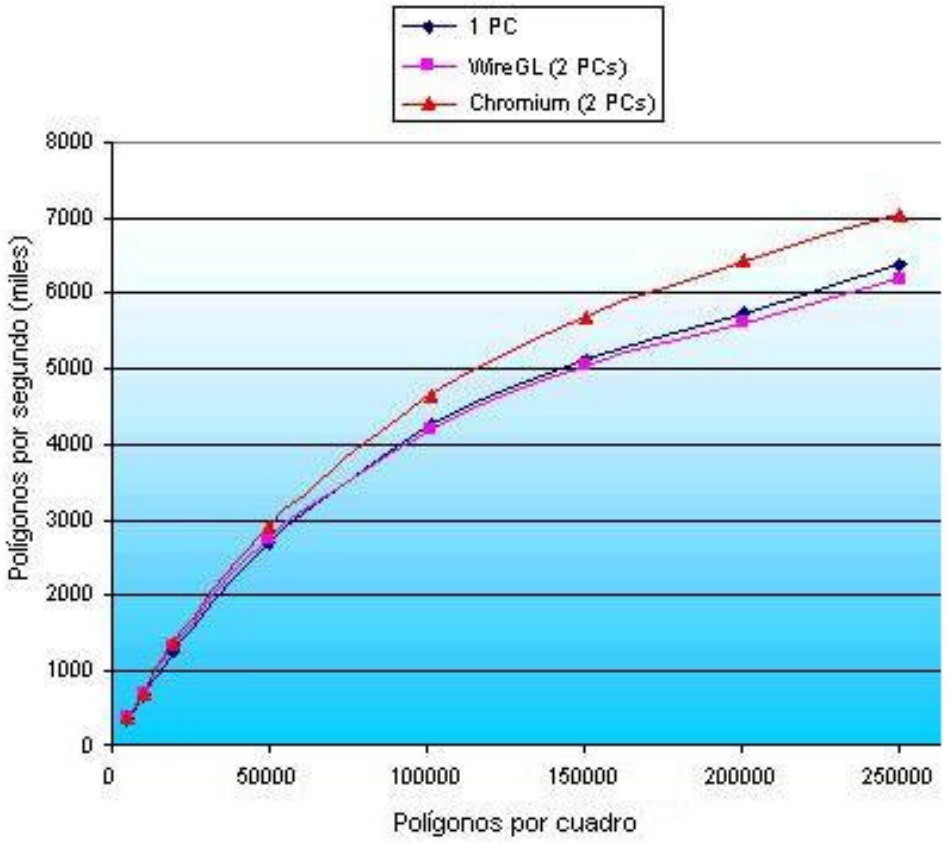


Figura 2-11 Comparación de Chromium y WireGL [65]

Además, como se puede ver en la Figura 2-11 de unas pruebas realizadas por el Sydney Vislab, Chromium es más rápido que WireGL. [65]

2.5 Software de visualización

Existen diversos paquetes de software que permiten la visualización de datos científicos. En esta sección se analizarán solamente paquetes que se pueden obtener gratuitamente y que permitan realizar visualizaciones climáticas.

La primera aplicación es OpenDX, que es un paquete que permite visualizaciones dentro de muchos campos (tales como medicina, economía, exploración petrolera, entre otras). La otra aplicación es Vis5d que es un paquete más especializado en visualizaciones climáticas.

2.5.1 OpenDX

El *Visualization Data Explorer* es una aplicación desarrollada por IBM desde 1991, que permite la visualización de datos, especialmente información 3D de simulaciones y datos obtenidos de observaciones científicas. [22]

En 1999, IBM abrió el código y liberó la licencia de este software. OpenDX no es más que la versión de código abierto del Visualization Data Explorer. [34]

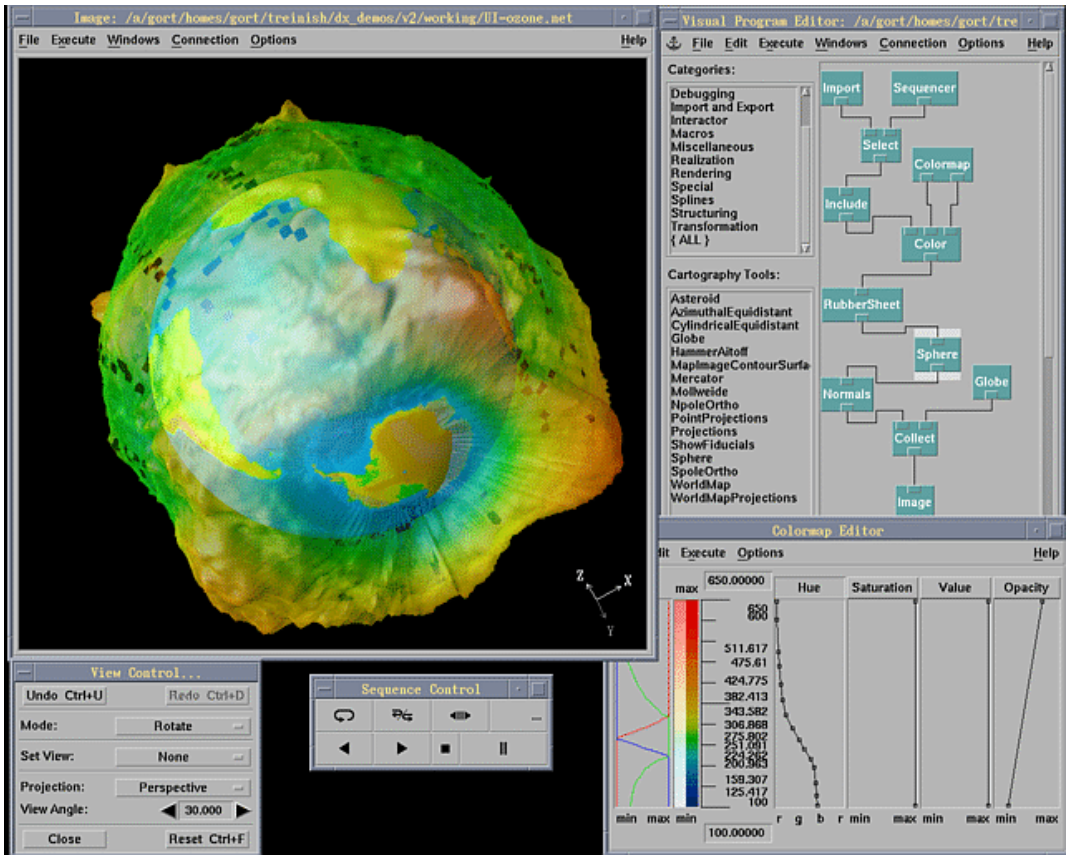


Figura 2-12 Ventana principal y visualización realizada con OpenDX [22]

OpenDX es un ambiente de visualización que permite al usuario aplicar técnicas avanzadas de análisis y visualización a un grupo de datos. Estas técnicas pueden ser aplicadas para ayudar al usuario a tener una mejor comprensión de datos de diversas áreas, tales como

la ciencia, ingeniería, medicina, química, exploración petrolera, meteorología, climatología, negocios, etc. [22]

OpenDX provee todas las herramientas necesarias para la manipulación, transformación, procesamiento, renderización y animación de datos. Además permite la visualización basada en puntos, líneas, áreas, volúmenes, imágenes, primitivas geométricas o cualquier combinación de estas. [22]

OpenDX soporta computadores de un solo procesador, de múltiples procesadores e incluso *clusters* de computadores. Este software puede ser instalado en múltiples sistemas operativos, incluidos Windows y Linux. [22]

OpenDX permite la renderización de imágenes por hardware, a través de OpenGL. Si un computador no tiene acelerador gráfico, las imágenes son renderizadas por software. [22]

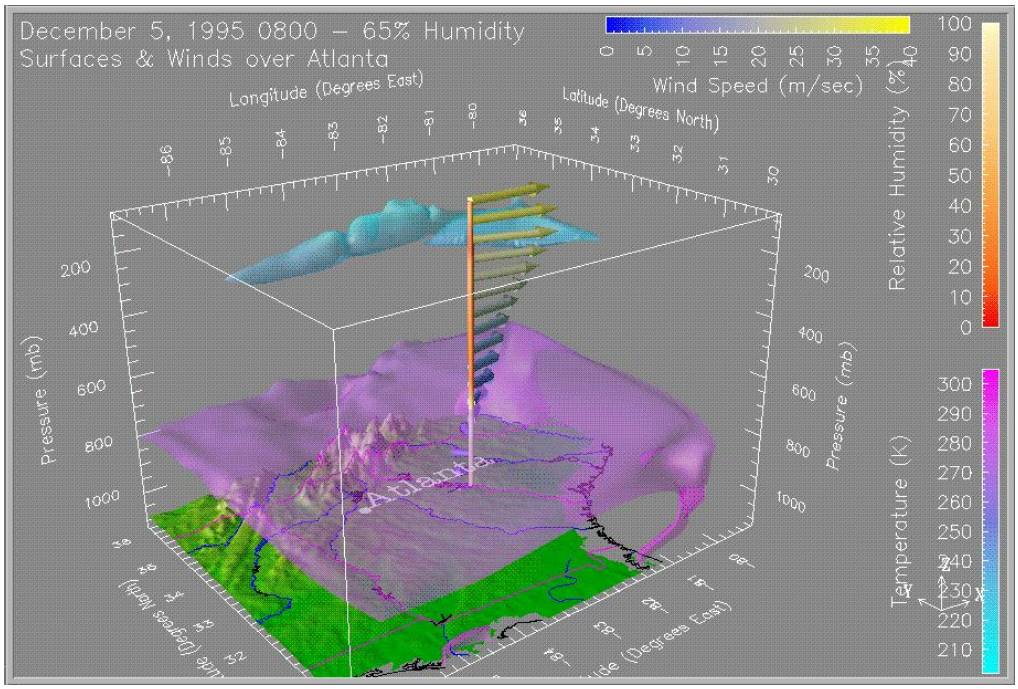


Figura 2-13 Visualización de condiciones climáticas [22]

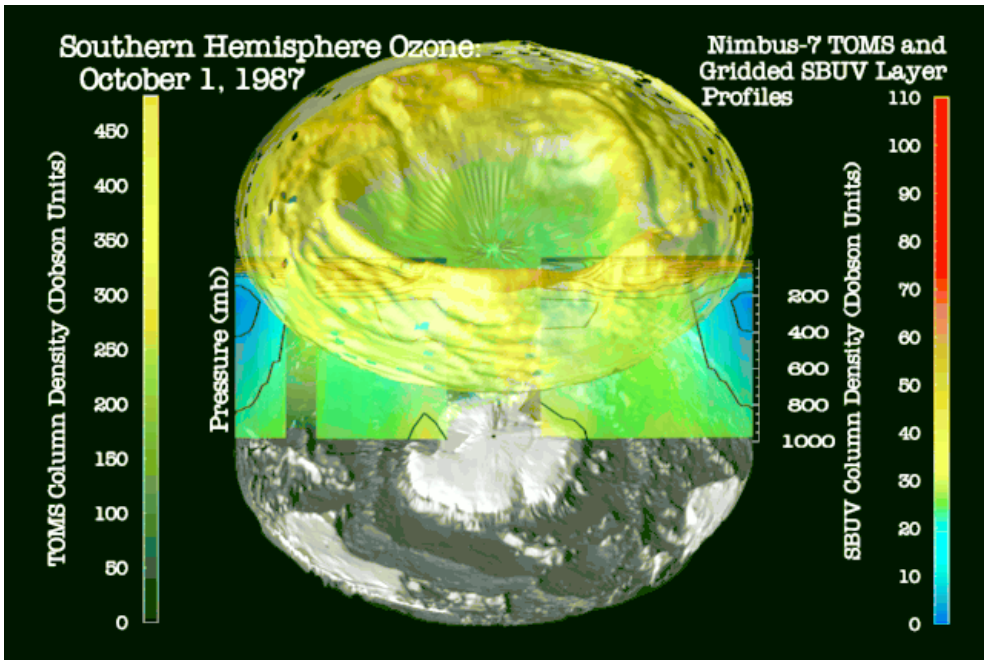


Figura 2-14 Visualización de la capa de ozono sobre el polo sur [22]

En la Figura 2-13 y la Figura 2-14, podemos ver ejemplos de visualizaciones meteorológicas realizadas con OpenDX. En la Figura 2-13, se muestra las condiciones climáticas pronosticadas para Atlanta durante las olimpiadas. En esta imagen se puede ver la relación de la temperatura con la humedad y la dirección del viento. En cambio en la Figura 2-14, se pude ver el agujero de la capa de ozono sobre la Antártica.

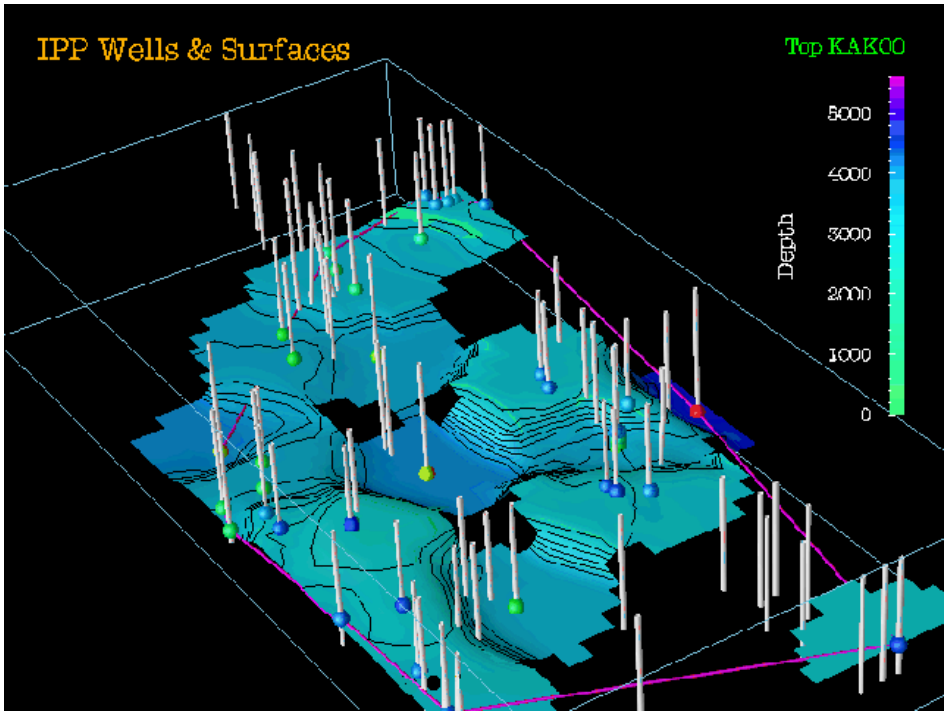


Figura 2-15 Visualización de un campo petrolero [22]

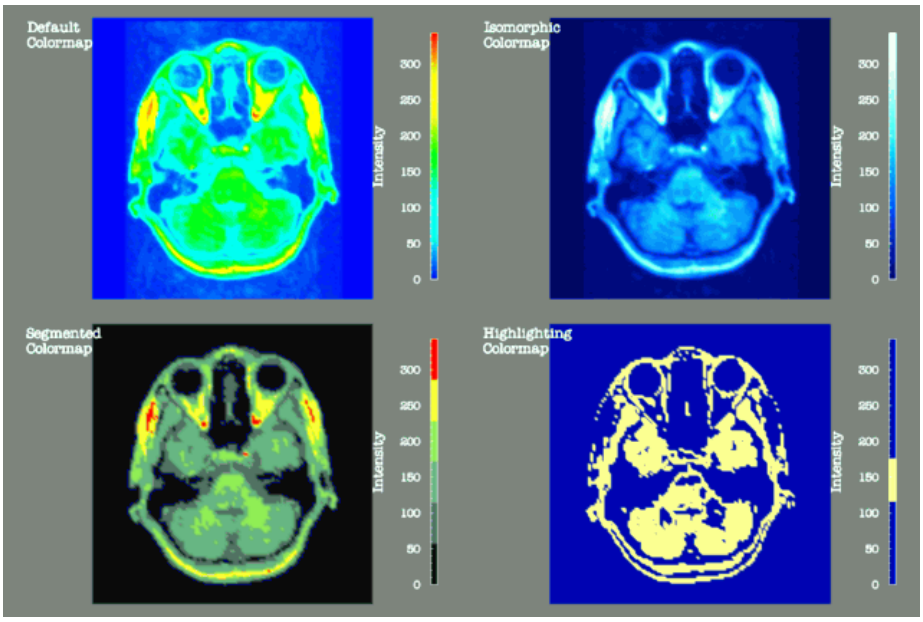


Figura 2-16 Visualización del cerebro [22]

En la Figura 2-15 y la Figura 2-16, podemos ver ejemplos de visualizaciones en otros campos, como la exploración petrolero y la medicina. En la Figura 2-15, se muestra la profundidad de diversos pozos petroleros en una zona. La Figura 2-16, en cambio, muestra una resonancia magnética del cerebro a la cual se le han aplicado diferentes combinaciones de colores.

2.5.2 Vis5d

Vis5d es un software desarrollado por el Proyecto de Visualización en el Centro de Ciencia Espacial e Ingeniería (SSEC) de la Universidad de Wisconsin-Madison. [56]

Vis5d es un sistema de software que permite la visualización de modelos climáticos numéricos. Este programa trabaja con datos en cinco dimensiones (las tres dimensiones de espacio, una dimensión de tiempo y otra para enumerar múltiples variables físicas). Este paquete permite la creación de superficies de contorno de tres dimensiones o superficies iso, curvas de nivel, cortes coloreados, etc. de información en una cuadrícula de tres dimensiones, en la cual se puede rotar o animar la imagen en tiempo real. [5]

Vis5d lee la información en dos formatos de archivos propios del programa, por lo tanto los datos deben ser convertidos a este formato para poder ser utilizado. [5]

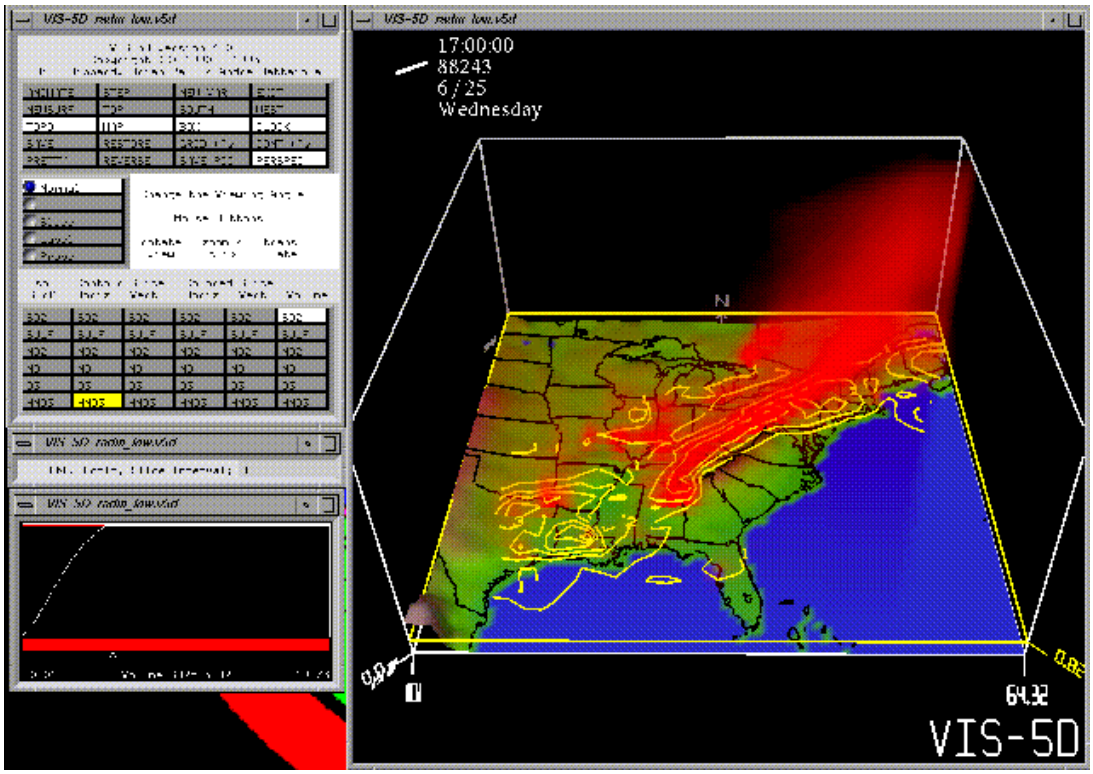


Figura 2-17 Pantalla principal de Vis5d [56]

La Figura 2-17, muestra la pantalla principal de Vis5d. Este programa tiene la capacidad de:

- rotar, acercar y alejar los gráficos;
- crear y mostrar trayectorias de viento;
- crear y reposicionar cortes verticales y horizontales;
- crear y editar etiquetas de texto;
- inspeccionar cuadrículas con el movimiento del ratón;
- crear animaciones;
- crear superficies de contorno de tres dimensiones (superficies iso);

- crear curvas de nivel y vectores de viento.

Vis5d trabaja con algunos sistemas, tales como SGI, Sun, Ultrix y Linux. La renderización 3D es realizada por el hardware si está disponible, en caso contrario la realiza por software usando Mesa, que es la librería de OpenGL en Linux. [5]

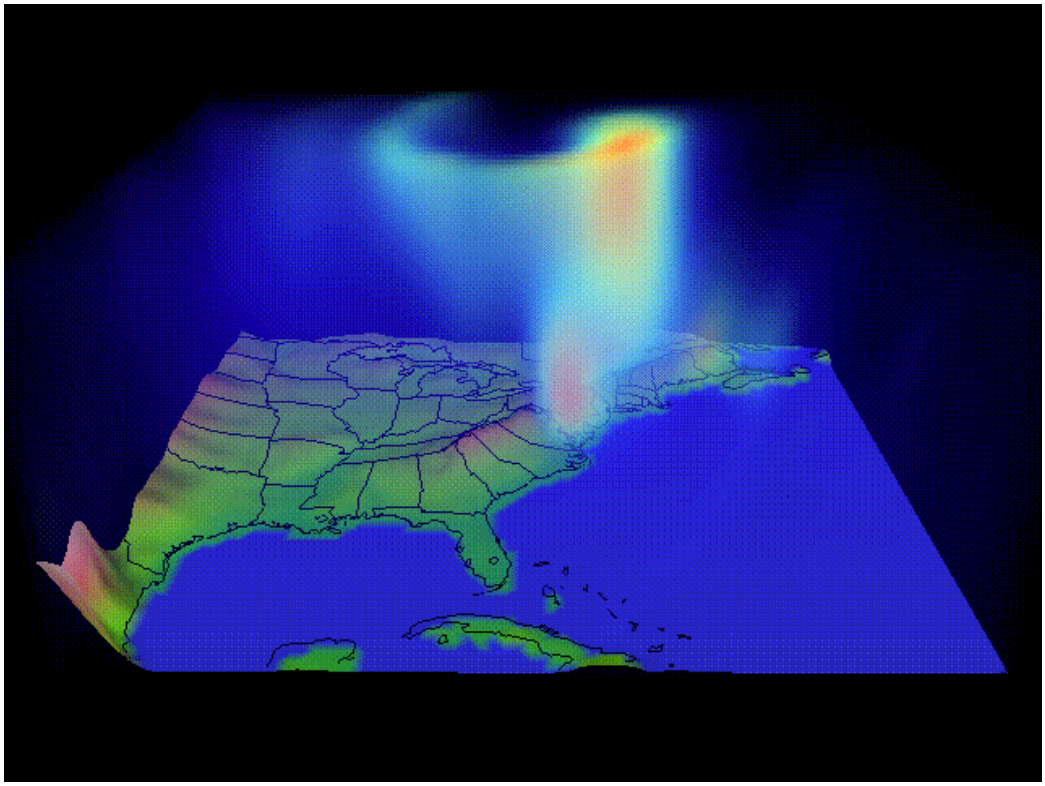


Figura 2-18 Visualización del viento en Vis5d [56]

La Figura 2-18 muestra una visualización del componente vertical del viento sobre los Estado Unidos durante una tormenta de invierno, realizada con Vis5d. La Figura 2-19 muestra una tormenta sobre Florida generada por colisiones de corrientes de aire.

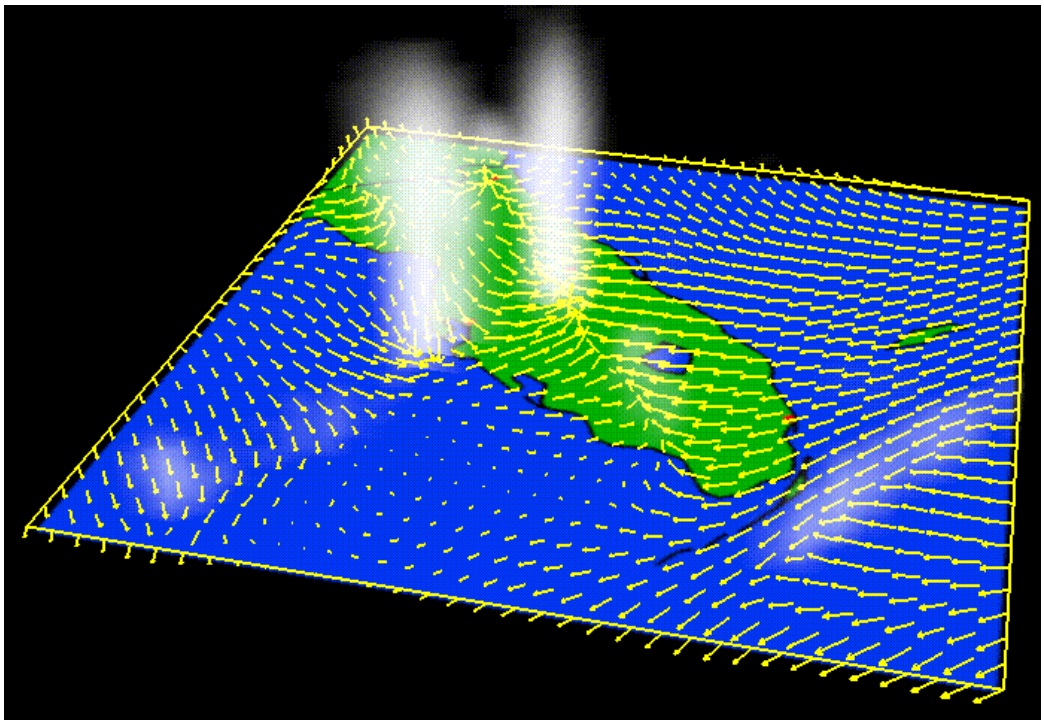


Figura 2-19 Visualización de una tormenta sobre Florida en Vis5d [56]

2.6 Software para colaboración remota

En esta sección se analizará VNC que es un programa que permite la colaboración y el control remoto de computadores.

2.6.1 VNC

Virtual Network Computing (VNC) es un software desarrollado por los Laboratorios AT&T de la Universidad de Cambridge. [60]

VNC es un software de control remoto que permite ver el escritorio de un ambiente de computación no solamente en el computador que está corriendo, sino desde cualquier computador que está conectado a Internet, sin importar la arquitectura del mismo. [60] Este software también permite que un escritorio sea accedido desde diferentes lugares simultáneamente por lo tanto permite compartir aplicaciones para trabajo cooperativo. [46]

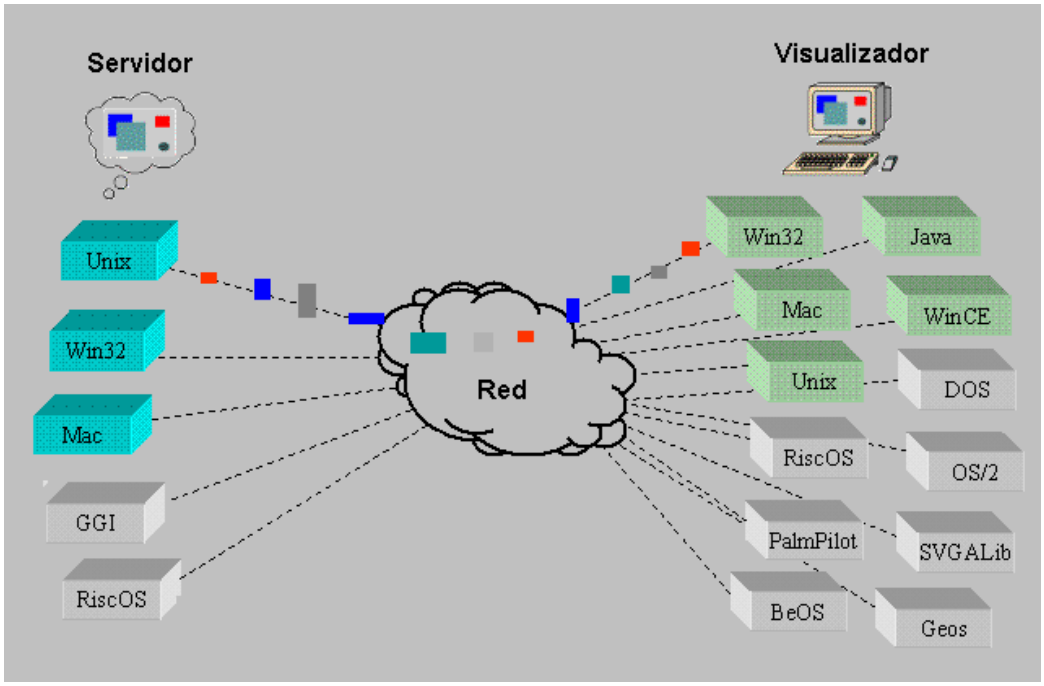


Figura 2-20 Diagrama de funcionamiento de VNC [44]

Como muestra la Figura 2-20, el servidor y el cliente o visualizador de VNC no tienen que tener la misma arquitectura ni estar corriendo el mismo sistema operativo. VNC puede correr en múltiples sistemas

operativos, tales como Windows, Linux, Solaris, etc. Además posee un visualizador Java lo que permite que el escritorio del servidor pueda ser visto por cualquier explorador que soporte Java. [44]

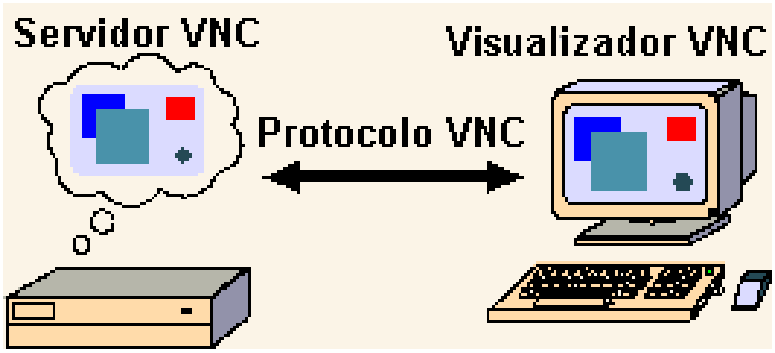


Figura 2-21 Arquitectura de VNC [46]

Como se puede ver en la Figura 2-21, la arquitectura de VNC es muy sencilla. El servidor VNC corre en un computador, el cliente o visualizador corre en otro computador, y esta recibe a través del protocolo VNC la información necesaria para mostrar en su pantalla lo que ocurre en el servidor. [46]

El protocolo VNC es un simple protocolo para acceso remoto a interfaces de usuario gráficas. Está basado en el concepto de *buffer* remoto. El protocolo simplemente permite al servidor actualizar el *buffer* con la información que debe ser mostrada en el visualizador. Este protocolo puede operar sobre cualquier protocolo de transporte confiable, tal como TCP/IP. [46]

Figura 2-22 Escritorio de Windows visto desde Linux con VNC [46]

La Figura 2-22 muestra el escritorio de un computador que tiene Windows como sistema operativo visto desde un computador que está corriendo en Linux.

2.7 Análisis de las alternativas y selección de la solución

En cuanto a las alternativas de hardware, a pesar de que un supercomputador daría un poder de procesamiento mucho mayor que cualquiera de las otras soluciones, su costo (millones de dólares) es

prohibitivo, por lo cual no es una opción factible para este proyecto. En cuanto a los computadores de alto rendimiento, si bien es cierto que su precio es más asequible, siguen siendo demasiado caras, como se mostró en la sección 2.1. En cambio, con un *cluster* de computadores podemos construir un sistema relativamente poderoso con una fracción de lo que invertiría en cualquiera de las otras dos opciones. El *cluster* puede ser construido con computadores tan baratos como el presupuesto lo limite, pero eso sí hay que dejar claro que mientras menos e inferiores computadores formen el *cluster*, el poder de procesamiento del mismo será menor.

Una vez que se ha decidido que se construirá un *cluster* de computadores, es necesario decidir que software se utilizará para la creación del mismo. Como se analizó en la sección 2.2, existen muchas opciones de software para realizarlo.

La primera opción (Beowulf), la descartamos porque a pesar de ser la más antigua, no ofrece paquetes gratuitos que faciliten la instalación, es decir habría que configurar todo prácticamente desde cero. Esto nos deja dos opciones, OSCAR y CLIC, que si ofrecen paquetes de instalación. La principal ventaja que posee CLIC sobre OSCAR es que además de ser relativamente fácil de instalar, es también fácil de configurar y monitorear la actividad del *cluster*. Además CLIC es un software desarrollado para cálculos masivos, por lo cual se ajusta más

a las necesidades de cómputo que puede tener el sistema. Por estas razones, el software que se utilizará para construir el *cluster* de computadores será CLIC.

En lo que se refiere al entorno gráfico, en la sección 2.3, se presentaron dos opciones, DMX y VNCWall. Como se indicó en esa sección, una de las principales desventajas de VNCWall es que necesita un computador extra que actúe exclusivamente como servidor, mientras que DMX no, lo cual elevaría el costo del sistema de visualización. Además, ese computador realiza todos los cálculos, mientras que los otros se limitan a mostrar solamente los píxeles que les son enviados. En cambio DMX si permite la distribución del procesamiento gráfico en todos los computadores del *cluster*. Otra ventaja importante de DMX es que cuando se manda a dibujar, no envía píxeles, sino primitivas de dos dimensiones, lo que hace que la transmisión de los datos sea mucho más rápida. Por todas estas razones, el software escogido para la distribución del entorno gráfico es DMX.

DMX lo único que hace es dividir una pantalla en varias partes e indicarle a cada proyector que parte de la misma le corresponde mostrar, pero en sí este software, aunque lo permite, no distribuye el procesamiento gráfico. Por esta razón es necesario que se utilice algún programa que distribuya este procesamiento.

En la sección 2.4, se analizaron dos paquetes que distribuyen instrucciones gráficas. Estos paquetes son WireGL y Chromium.

Como se indicó, Chromium es la continuación del proyecto que empezó con WireGL, por esta razón Chromium ofrece múltiples ventajas. Una de las principales, es que se han implementado muchas más librerías de OpenGL en Chromium que en WireGL, esto significa que es muy probable que cualquier programa de visualización que utiliza OpenGL pueda correr en Chromium. En cambio, muchos programas de visualización actuales, inclusive algunos de los que se van utilizar para las visualizaciones climáticas, como OpenDX y Vis5d, no tienen todas las librerías necesarias implementadas en WireGL, por lo que no funcionan.

Además, como se demostró mediante algunas pruebas realizadas por el Sydney Vislab, Chromium es más rápido que WireGL.

Por estas razones, para la distribución en paralelo de las imágenes gráficas, se utilizará Chromium.

Para la visualización propiamente dicha de las imágenes climatológicas, se utilizarán dos paquetes de software que fueron detallados en la sección 2.5.

El primer programa es OpenDX, que es una poderosa herramienta que permite la visualización de datos de múltiples campos diferentes tales como la medicina, la exploración petrolera, la física, entre otros. Pero

que también tiene capacidad de crear visualizaciones y animaciones muy detalladas de diversos fenómenos climáticos.

De igual manera, se utilizará Vis5d, que es un software especializado en la visualización de modelos climáticos, y que también permite la creación de animaciones y visualizaciones en tres dimensiones.

En la sección 2.6, se analizó VNC, que es el software que permitirá la colaboración remota del sistema con otras unidades de investigación dentro de la universidad o alrededor del mundo. Este software permite el acceso y control de computadores remotamente sin que ellos estén utilizando el mismo sistema operativo, por lo tanto, a pesar que el sistema correrá sobre Linux, tranquilamente puede acceder a máquinas que tengan Windows o MacOS como sistema operativo.

CAPÍTULO 3

3 DISEÑO

En este capítulo se elabora el diseño lógico y físico del *cluster* de computadores. Además, describe el diseño del sistema distribuido de renderización de gráficos. Por último se especifica el diseño del sistema de proyección y los muebles necesarios para su soporte.

3.1 Diseño general

De manera general, el sistema de visualización, como se puede ver en la Figura 3-1, se puede dividir en tres partes principales:

- sistema de *cluster*,
- sistema distribuido de renderización, y,
- sistema de proyección y pantalla.

En cuanto al funcionamiento, el sistema de visualización trabajará de la siguiente manera. El usuario se comunica directamente con la aplicación que quiere correr, esta a su vez envía los requerimientos necesarios al sistema distribuido de renderización. Tanto las aplicaciones como el sistema de renderización corren sobre un *cluster* de computadores, pero esto es transparente para el usuario, porque él tendrá la impresión de estar trabajando solamente con un computador.

El sistema de renderización se encarga de dividir las órdenes y determinar el destinatario de cada una. Luego las envía a un sistema de proyección.

El sistema de proyección muestra los resultados de los cálculos en una pantalla para que puedan ser vistos por el usuario, completando el proceso.

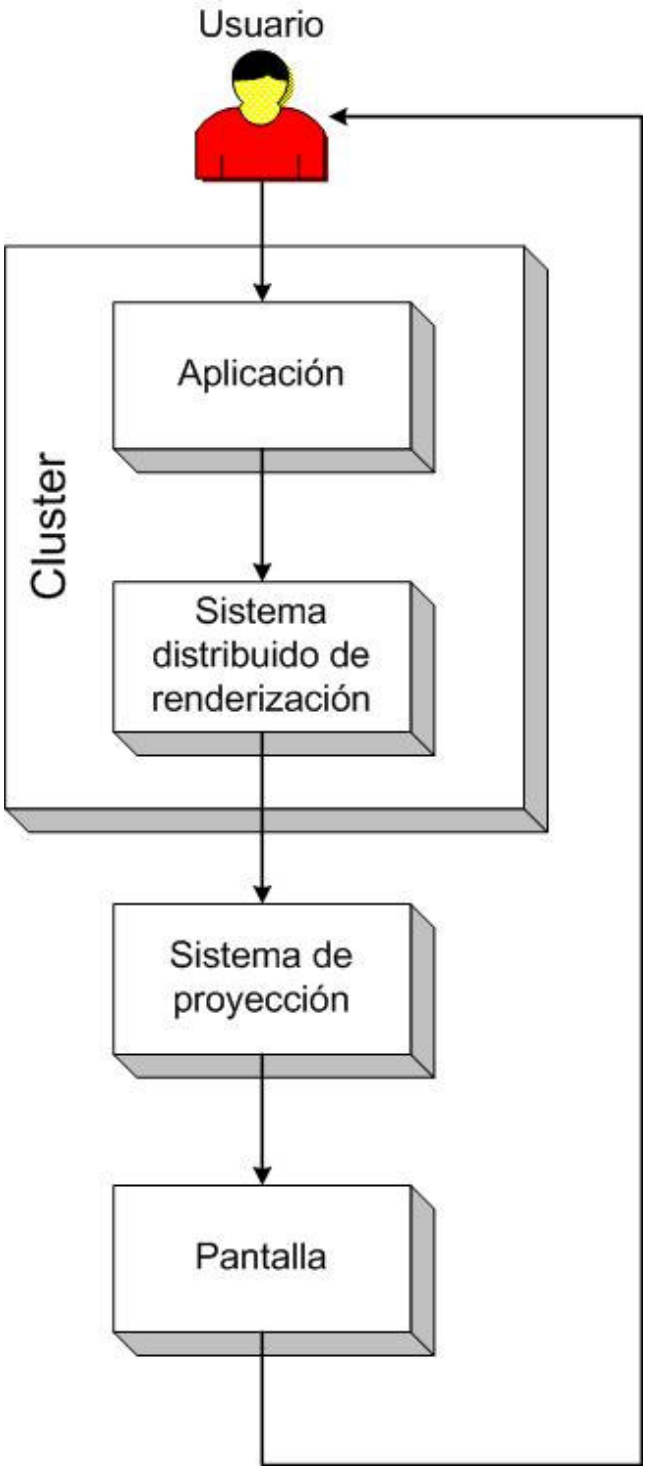


Figura 3-1 Diagrama general

En cuanto a la parte física o hardware del sistema, como se puede apreciar en la Figura 3-2, el usuario se comunicará con el servidor mediante dispositivos de entrada comunes, tales como el teclado y el ratón.

El servidor del *cluster* estará conectado con los demás nodos a través de un *switch*. Cada computador tendrá una tarjeta de red para poder enlazarse al *switch* mediante cables de red.

Además, tanto los nodos como el servidor, tendrán tarjetas aceleradoras de vídeo. Estas tarjetas servirán para que se conecten a sus respectivos dispositivos de salida, que en el caso de este sistema, serán proyectores digitales.

Cada proyector tendrá asignada una sección definida de la pantalla, por lo que el computador que este conectado a un proyector en particular, será el responsable de la parte de la pantalla a la cual proyecta el mismo.

Tanto la pantalla como los proyectores contarán con un sistema de soporte. El sistema de soporte de los proyectores debe permitir la mayor cantidad de grados de libertad posible, de tal manera que la alineación de los mismos no resulte demasiado complicada.

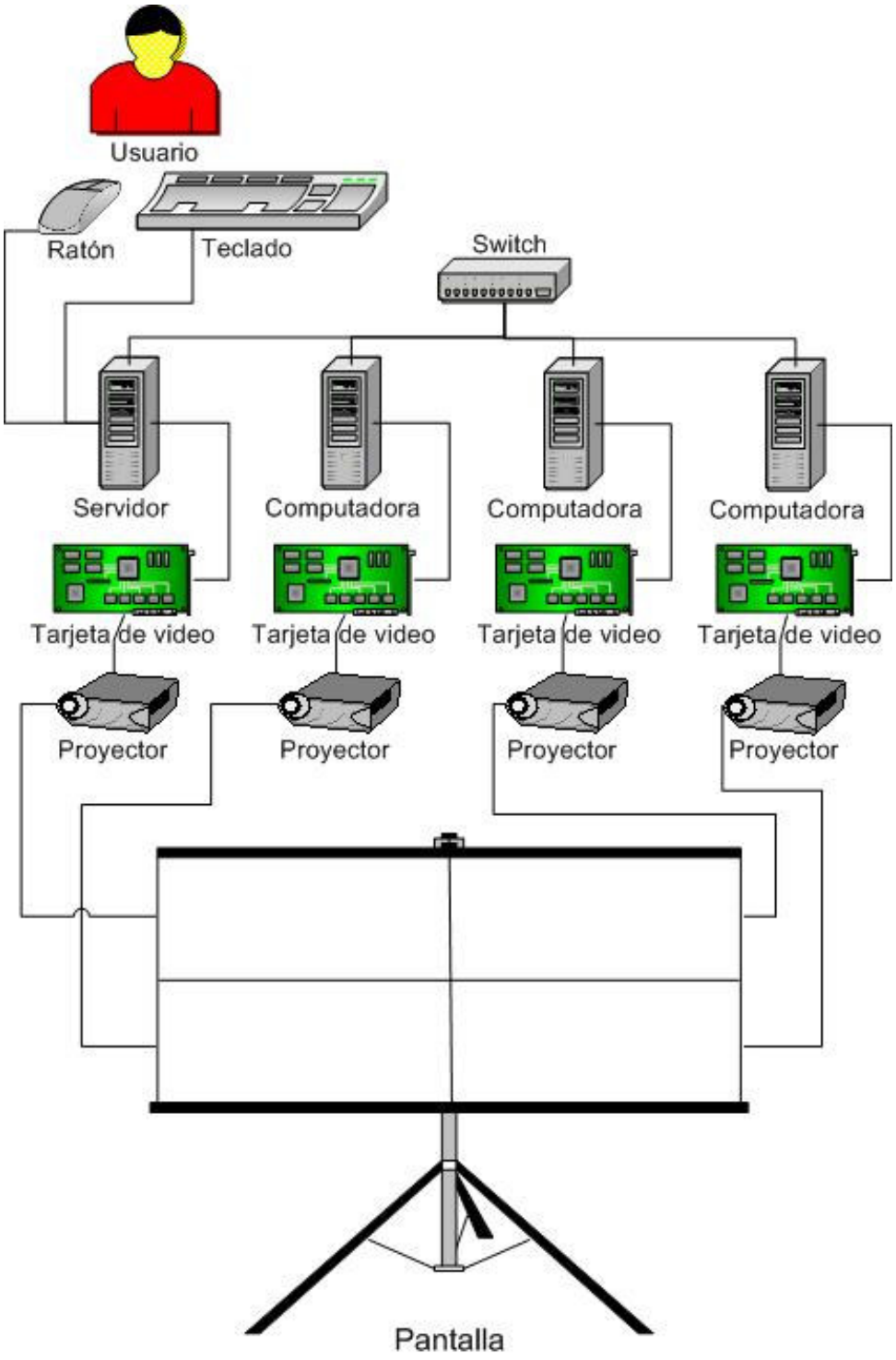


Figura 3-2 Diagrama físico general

3.2 Sistema de *cluster*

3.2.1 Diseño lógico

El *cluster* va a estar formado por cuatro computadores. Uno de ellos actuará como servidor, y los otros tres serán nodos.

El servidor se encargará de proveer los siguientes servicios:

- Verificación de usuarios, permitiendo el acceso a los recursos del *cluster* solamente a los usuarios autorizados.
- Configuración de *host* dinámico (DHCP), es decir asignación automática de direcciones IP.
- Servicio de nombre de dominio (DNS), el cual traduce los nombres de dominio a direcciones IP.
- Sistema de archivos compartidos (NFS), el cual permite a usuarios de una red acceder carpetas y archivos compartidos en cualquier computador que pertenece a la misma.
- Sistema de información de la red (NIS), el cual permite a los computadores de una red compartir la información de configuración. Los archivos de administración se guardan en el servidor NIS y son accedidos por los clientes, en lugar de que cada uno guarde sus propios archivos de administración.

- Montado automático de los clientes, el cual se realiza a través de *autofs*, que permite el montado automático de sistemas de archivos cuando son utilizados, y los desmonta después de un cierto periodo de inactividad.
- Sincronización del tiempo del sistema, la cual se realiza mediante el *ntpd*, que es un protocolo utilizada para sincronizar los relojes de los computadores de una red.
- Comunicación de nodo a nodo, la cual se realiza a través de *rsh*, que es un comando de red que permite ejecutar un comando cualquiera en un *host* remoto, pasando las entradas y recibiendo las salidas.
- Ejecución paralela remota, la cual se realiza mediante *gexec*, que es un sistema de ejecución remota que distribuye el trabajo y el uso de recursos en un *cluster*.
- Monitoreo del rendimiento del *cluster*, para el cual se utiliza *Ganglia*, que es un sistema que permite la visualización del desempeño de sistemas computacionales distribuidos.

La comunicación entre el *cluster* y el usuario se realizará a través del servidor, de tal manera que el mismo tenga la impresión de estar trabajando solo con un computador. El servidor se encargará de repartir las instrucciones para que el procesamiento sea más rápido, pero esto será transparente para el usuario.

3.2.2 Diseño físico

Físicamente, el *cluster* estará constituido por cuatro computadores que posean una tarjeta de red. Los computadores se conectaran entre sí a través de un *switch*.

Cabe indicar que es necesario utilizar un *switch* y no un *hub*. Un *hub* actúa solamente como un concentrador, por lo que cuando se manda un paquete de información, lo envía a todos los computadores conectados al aparato. Si un computador intenta enviar un nuevo paquete al mismo momento en que está recibiendo otro, ocurre una colisión y se deberán reenviar los paquetes, por lo que la latencia del sistema aumenta.

En cambio, el *switch* almacena las direcciones MAC de cada computador, de tal manera que el paquete es enviado solamente al computador de destino. Esto evita el envío de información innecesaria a través de la red, disminuyendo la probabilidad de que ocurran colisiones, y por consiguiente la latencia del sistema disminuye.

El tiempo que se demora el sistema en enviar los datos a cada computador es muy importante debido a los grandes volúmenes de información que se necesitan transmitir para realizar visualizaciones científicas.

Como se indicó en la sección 3.2.1, la comunicación entre el *cluster* y el usuario se realizará solamente a través del computador que actúa como

servidor, por lo tanto los dispositivos de entrada (teclado y ratón) y de salida deben de estar conectados a este, como se puede apreciar en la Figura 3-3.

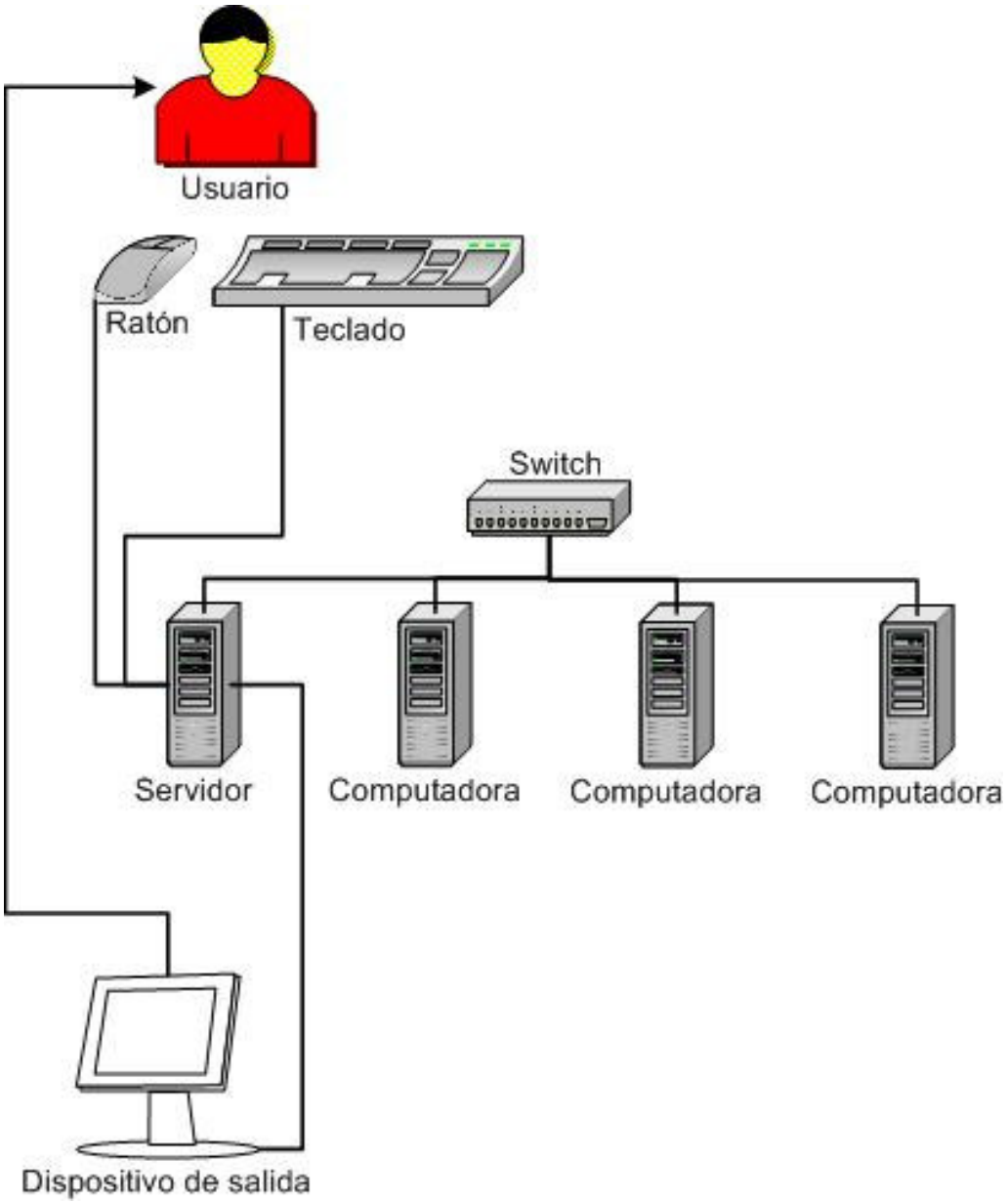


Figura 3-3 Diagrama físico del *cluster*

3.2.3 Sistema distribuido de renderización

En un sistema de visualización se utilizan gran cantidad de gráficos e imágenes de alta resolución, lo más importante para que el sistema funcione a una velocidad adecuada es que la renderización de estas imágenes se realice rápidamente.

Aprovechando el *cluster* de computadores que posee el sistema, la renderización de imágenes se realizará de forma distribuida.

La distribución de las instrucciones gráficas se realizará de forma diferente dependiendo de si estas son en dos o tres dimensiones. Cuando una aplicación necesite una librería de dos dimensiones, esta se comunicará directamente con el sistema de distribución de escritorio (en el caso de este sistema de visualización es DMX), el cual a su vez enviará las instrucciones al servidor X de cada computador del *cluster*, indicando que sección del escritorio le corresponde mostrar a cada una de ellas.

En el caso de que se necesiten librerías de tres dimensiones, la aplicación se comunicará primero con el sistema de distribución de operaciones OpenGL (en el caso de este sistema es Chromium). Una vez que se hayan realizado todos los cálculos para la creación del gráfico, Chromium enviará los resultados al sistema de distribución de

escritorio, el cual completará el proceso hasta enviar las instrucciones al sistema de proyección, como se puede ver en la Figura 3-4.

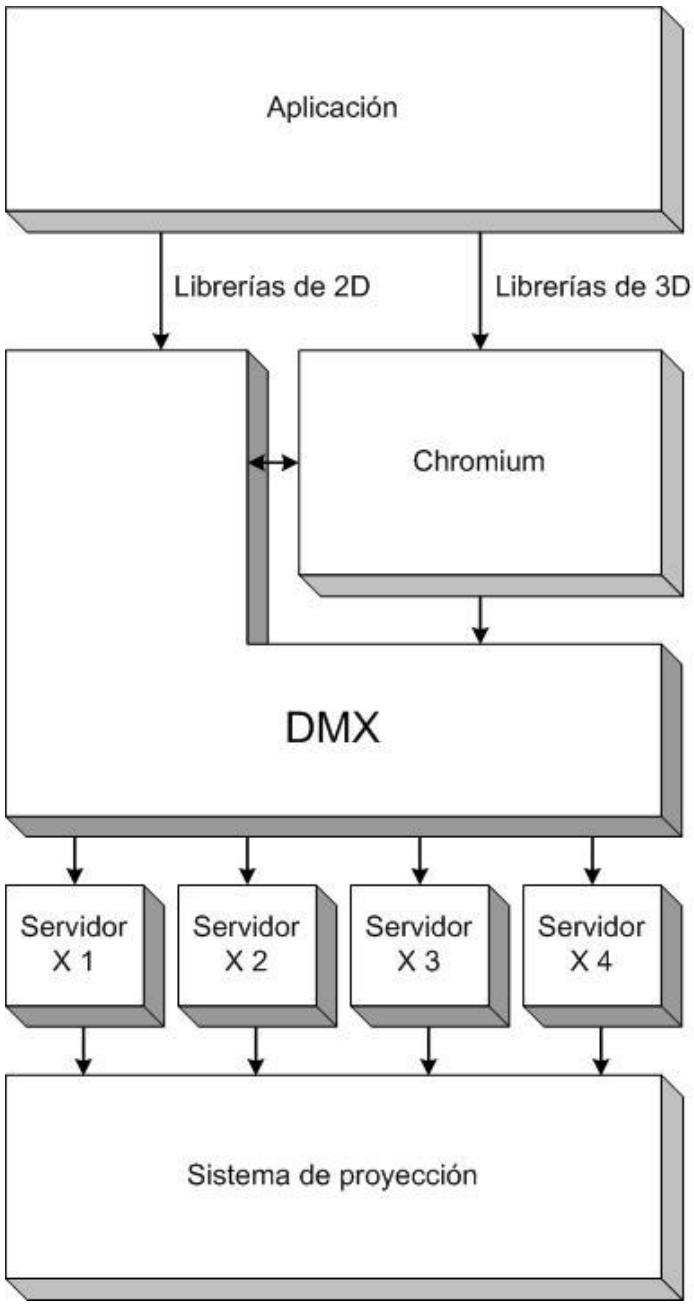


Figura 3-4 Diagrama del sistema distribuido de renderización

3.2.4 Sistema de distribución de escritorio

Para la distribución del escritorio o *desktop*, se utilizará el paquete de software DMX. Para este sistema en particular, la configuración con la que debe trabajar DMX se muestra en la Figura 3-5.

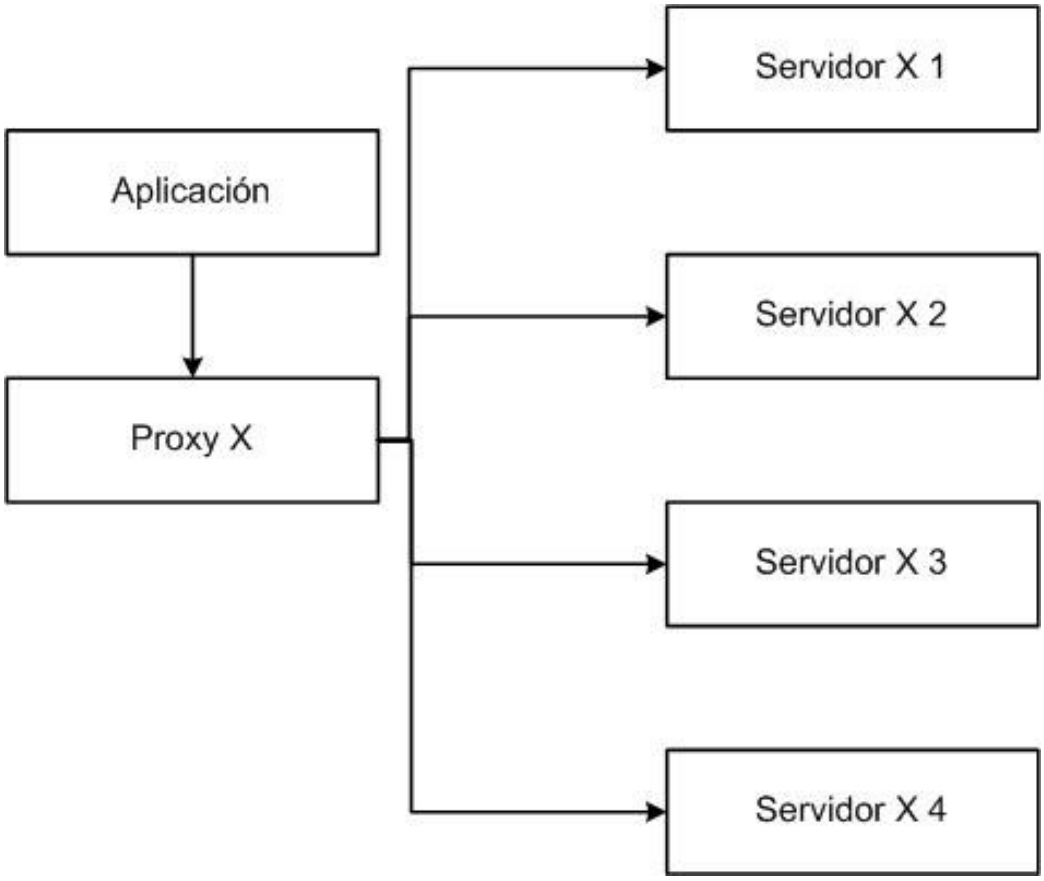


Figura 3-5 Diagrama de funcionamiento de DMX

El computador que actúa como servidor, tendrá un "*proxy* X," llamado *Xdmx*. La aplicación se comunicará con este *proxy*, y él se encargará de determinar que parte del escritorio le corresponde mostrar a cada uno de los servidores X.

En este sistema, el computador que actúa como *proxy* también será un servidor X de salida, de tal manera que solamente se necesitarán cuatro computadores, pues no será necesario utilizar una quinta solamente para la distribución de instrucciones.

Se utilizará *Xinerama*, que es una extensión X que permite que múltiples pantallas físicas sean controladas por un solo servidor X como si fueran una sola pantalla.

Los dispositivos de entrada serán manejados por el mismo computador que actúa de *proxy*, por lo tanto deberán ser conectados directamente a este computador y no podrán ser utilizados por otro servidor X.

Los requerimientos de renderización serán aceptados por el *proxy*, y este determinará que parte de la pantalla se ve afectada por el pedido de renderización y enviará el requerimiento a los computadores que les corresponda a través de llamadas de la librería X11.

Los demás requerimientos, tales como los de movimiento o cambio de tamaño de ventanas, apertura o cierre de programas, entre otros, serán manejados por el computador que actúa de *proxy* y este solo enviará los

resultados de estos requerimientos a los nodos si la parte del escritorio que les corresponde se ven afectados por los mismos.

3.2.5 Sistema de distribución de operaciones OpenGL

Para la distribución de operaciones OpenGL se utilizará Chromium. Como se indicó en la sección 2.4.2, Chromium permite la renderización interactiva en *clusters* de computadores.

Este programa se encargará de interceptar las llamadas OpenGL que haga cualquier aplicación, y distribuirá las operaciones de creación de gráficos de tres dimensiones entre los computadores que conforman el *cluster*, tal como se ve en la Figura 3-6.

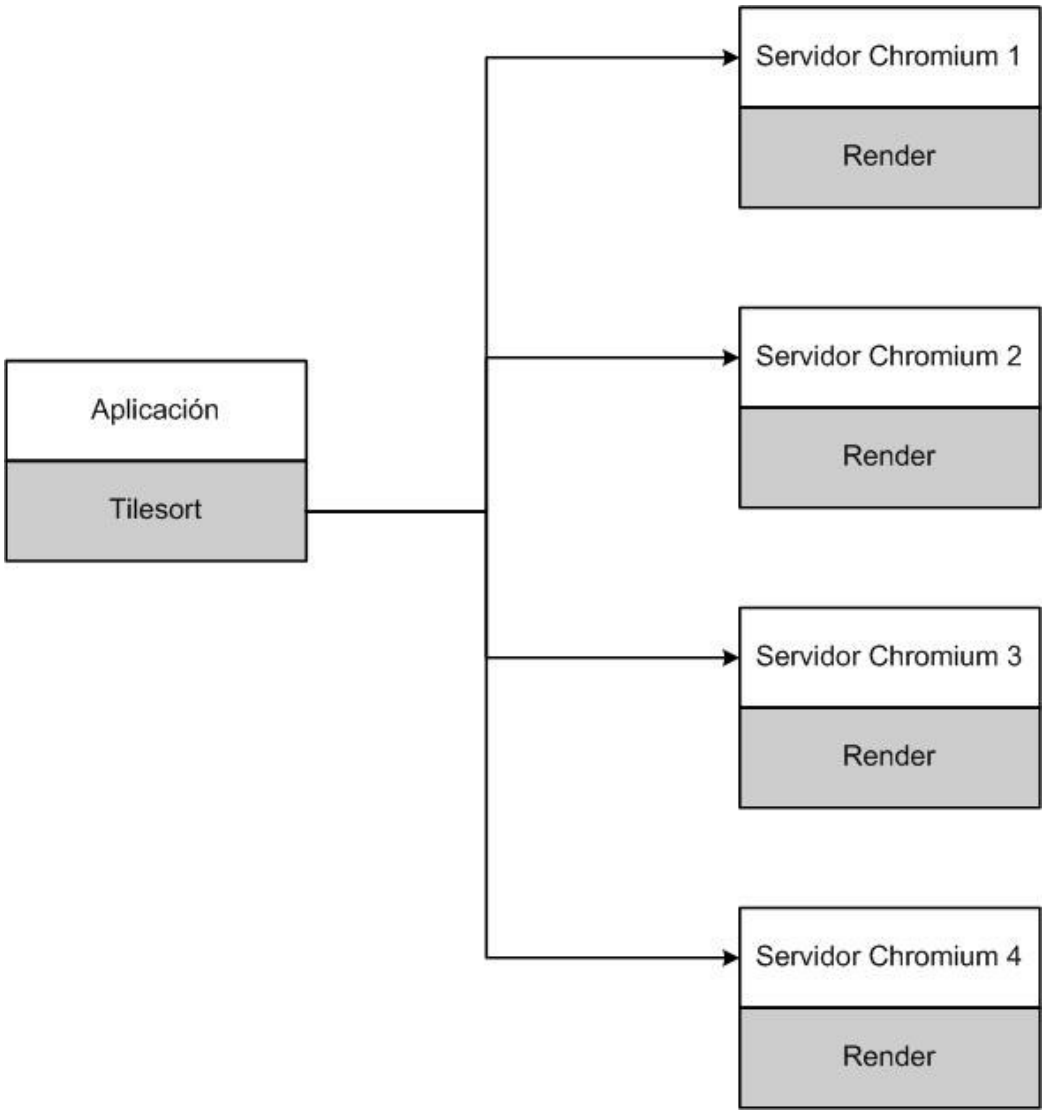


Figura 3-6 Diagrama de funcionamiento de Chromium

En este sistema, Chromium estará configurado para utilizar un procesador de cadenas *sort-first* llamado *tilesort*. Esto quiere decir que, antes enviar la información a ser procesada por cada computador, el

gráfico que se quiere mostrar es dividido en cuatro partes que no se superponen, y se le asigna una sección a cada uno.

Una vez que todas los computadores hayan terminado de realizar los cálculos necesarios para mostrar la imagen, Chromium se comunicará con DMX para que el nodo sepa si le corresponde mostrar la información que calculó, si es así lo realizará. Si no, enviará a través de la red los resultados al computador que le corresponde mostrarlos.

Los principales componentes que utilizará Chromium para la distribución de las operaciones gráficas son:

- El *mothership*, que es el nodo que está encargado de las solicitudes de configuración para todos los componentes del sistema. En el caso de este sistema el *mothership* será el servidor del *cluster*.
- Las unidades de renderización o servidores Chromium, que realizarán los cálculos necesarios para satisfacer las llamadas OpenGL de las aplicaciones. En el caso de este sistema, tanto el servidor del *cluster* como los nodos funcionarán como servidores Chromium.
- Un *script* de configuración, el cual determinará cuantos servidores Chromium hay en el sistema, los nombres de estos servidores y la sección de la pantalla que debe calcular cada uno.

- Una librería “impostora,” la cual debe ser utilizada por las aplicaciones en lugar de las librerías OpenGL. Esta librería deberá tener implementadas todas las funciones que cualquier programa podría solicitar a las librerías OpenGL. Cada servidor Chromium debe tener acceso a esta librería “impostora.” En este sistema, la librería se encontrará en una carpeta compartida del servidor del *cluster*, la cual puede ser accesada por todos los nodos.
- El lanzador de la aplicación o *crappfaker*, el cual corre las aplicaciones de tal forma que estas cargan la librería “impostora” de OpenGL en lugar de las librerías del sistema. En el caso de este sistema, el *crappfaker* correrá en el servidor del *cluster*.

3.3 Sistema de proyección

Para la proyección se utilizarán cuatro proyectores con resolución nativa de 800 x 600, de tal manera que se tendrá una pantalla con una resolución total de 1600 x 1200.

Los proyectores serán de 1,500 lúmenes para que no sea necesario que la habitación se encuentre totalmente oscura al momento de utilizar la pantalla.

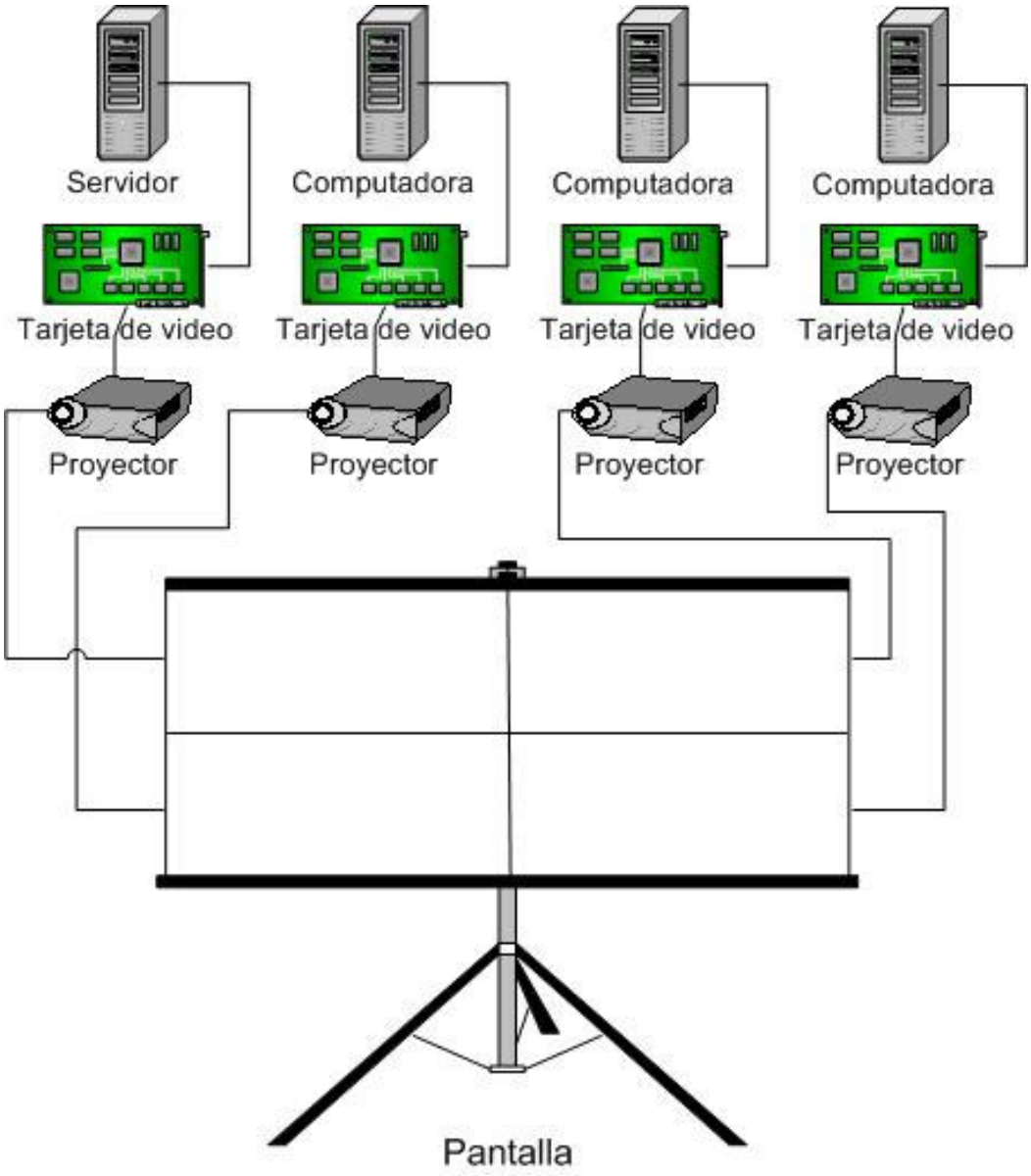


Figura 3-7 Diagrama del sistema de proyección

Como se puede ver en la Figura 3-7, cada proyector estará conectado a un computador a través de una tarjeta de vídeo. Además, cada uno tendrá asignado una sección de la pantalla de tal forma que sin

superponerse formarán una pantalla de la misma proporción (4:3) que un monitor normal. Es decir, dos proyectores se encargarán de la parte superior de la pantalla, y los otros dos, de la parte inferior.

Para mostrar las imágenes enviadas por los proyectores se utilizará una pantalla. El tamaño de la misma estará limitado por el tamaño de la habitación, tomando en cuenta que se deben dejar al menos cuarenta centímetro entre el suelo y la pantalla para que no sea incómodo para las personas mirarla.

Como se puede ver en la Figura 3-8, se utilizará una pantalla de proyección trasera por las múltiples ventajas que ofrece, tales como el hecho de que el expositor se puede parar delante de la proyección sin cubrirla. Además, esto permitirá que todo el sistema esté oculto detrás de la pantalla, con lo cual se facilitará que los usuarios tengan la impresión de estar trabajando con un solo computador.

Para que el usuario de la pantalla pueda manejarla desde el frente, se utilizarán un teclado y un ratón inalámbrico, cuyo receptor estará conectado al computador que actúa como servidor del *cluster*.

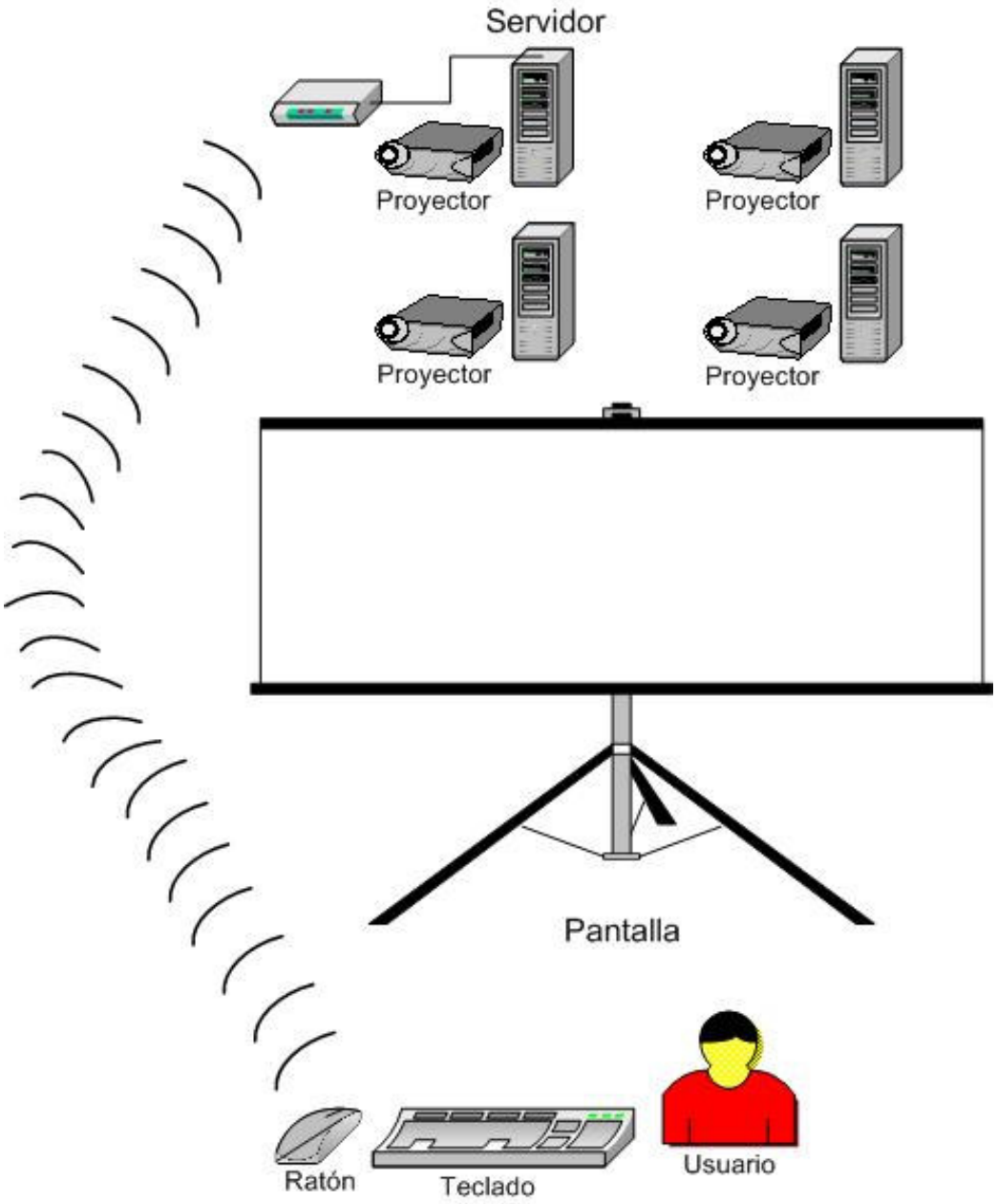


Figura 3-8 Diagrama de posición de la pantalla

3.3.1 Sistema de soporte

Para poner los proyectores y los computadores se utilizará un mueble dividido en cuatro compartimientos, como el que se puede ver en la Figura 3-9.

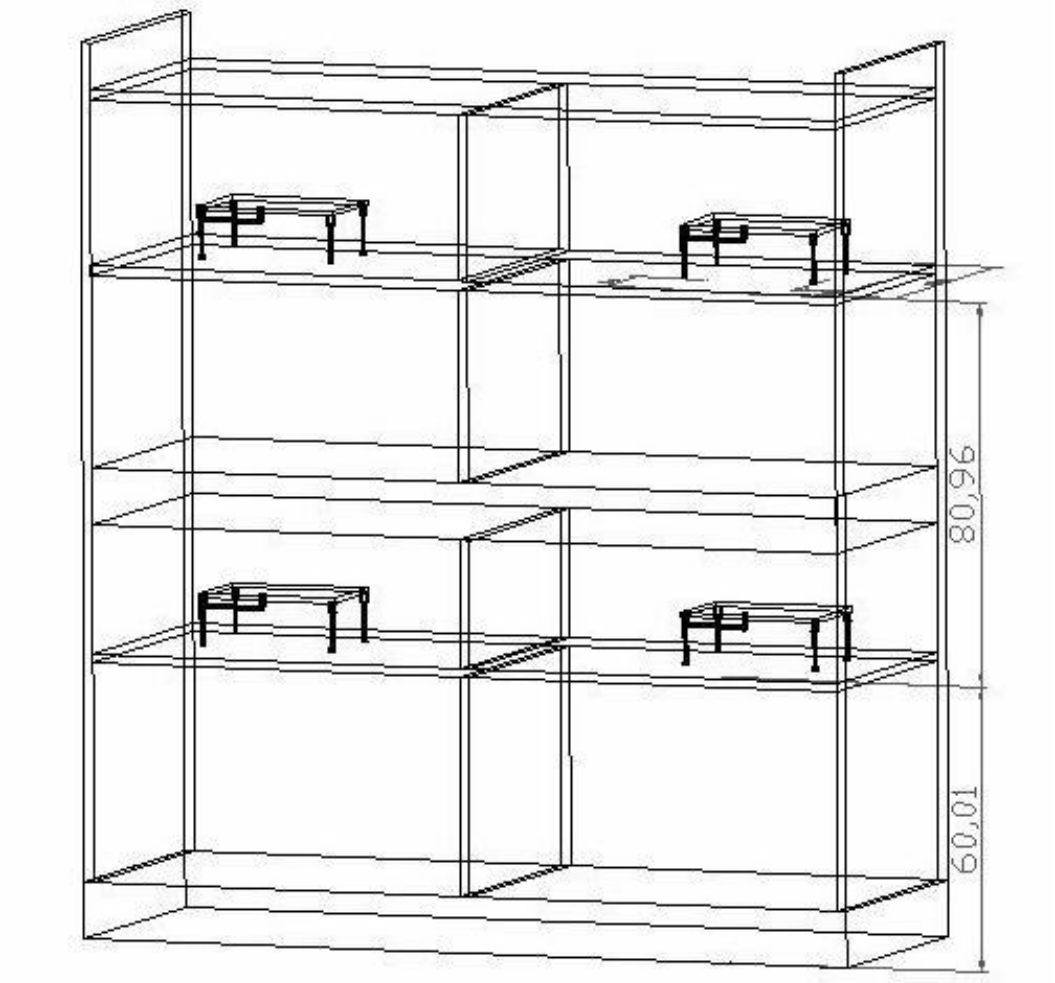


Figura 3-9 Mueble para los proyectores y los computadores

En cada compartimiento se pondrán un computador y un proyector, con su respectiva base.

Las bases para los proyectores, que se muestran en la Figura 3-10, se colocarán sobre planchas metálicas.

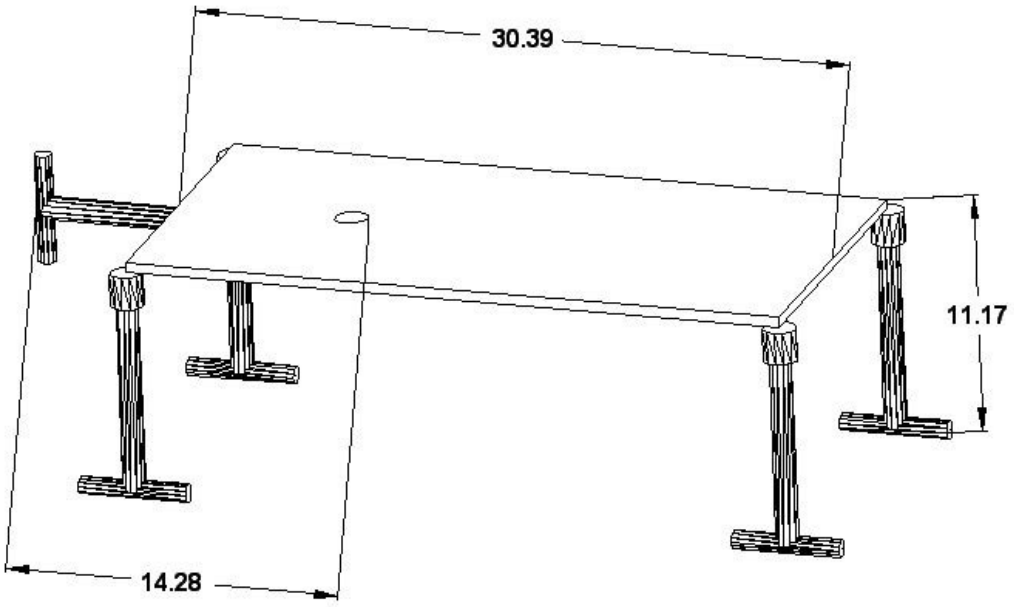


Figura 3-10 Base para proyector

Estas bases tendrán tornillos de precisión, los cuales servirán para mover los proyectores tanto de forma vertical como horizontal. Estos tornillos permitirán realizar ajustes finos en el momento que se tengan que cuadrar y alinear los proyectores.

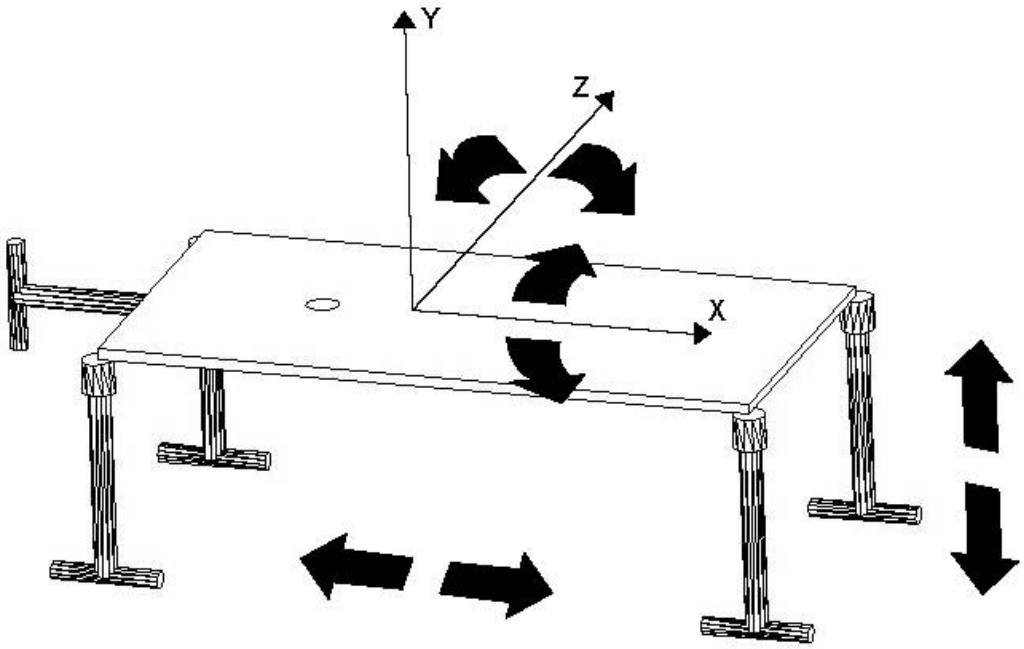


Figura 3-11 Grados de libertad que provee la base de los proyectores

Como se muestra en la Figura 3-11, la base del proyector provee cuatro grados de libertad, ya que además de poder moverse de forma vertical y horizontal, los proyectores también pueden inclinarse tanto hacia arriba o hacia abajo, como hacia los costados, utilizando los cuatro tornillos de la base.

También se utilizará una base para sostener la pantalla como la que se observa en la Figura 3-12.

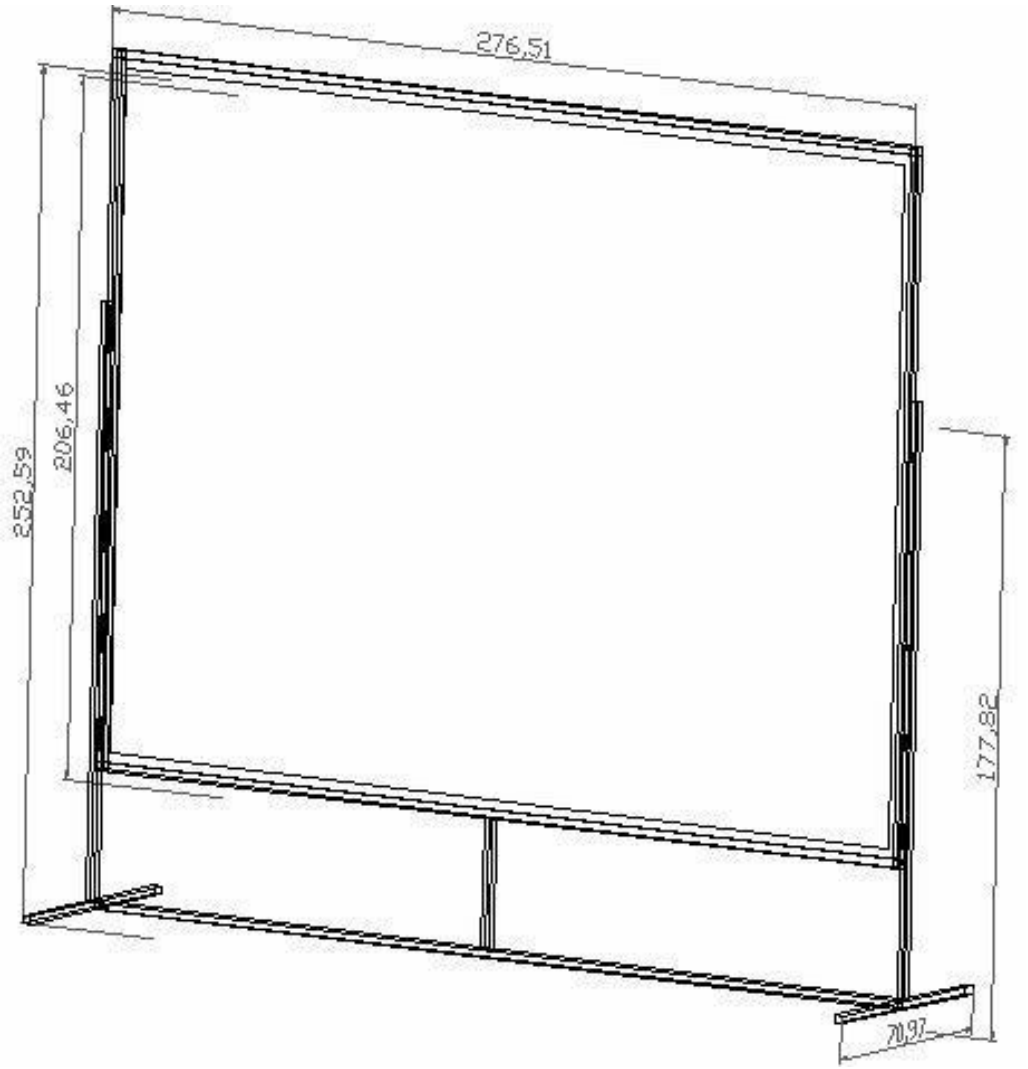


Figura 3-12 Pantalla con su base

Como se indicó en la sección anterior, se utilizará una pantalla de proyección trasera, por lo que la distribución del sistema de soporte en la sala utilizada para el sistema de visualización deberá quedar como se ve en la Figura 3-13.

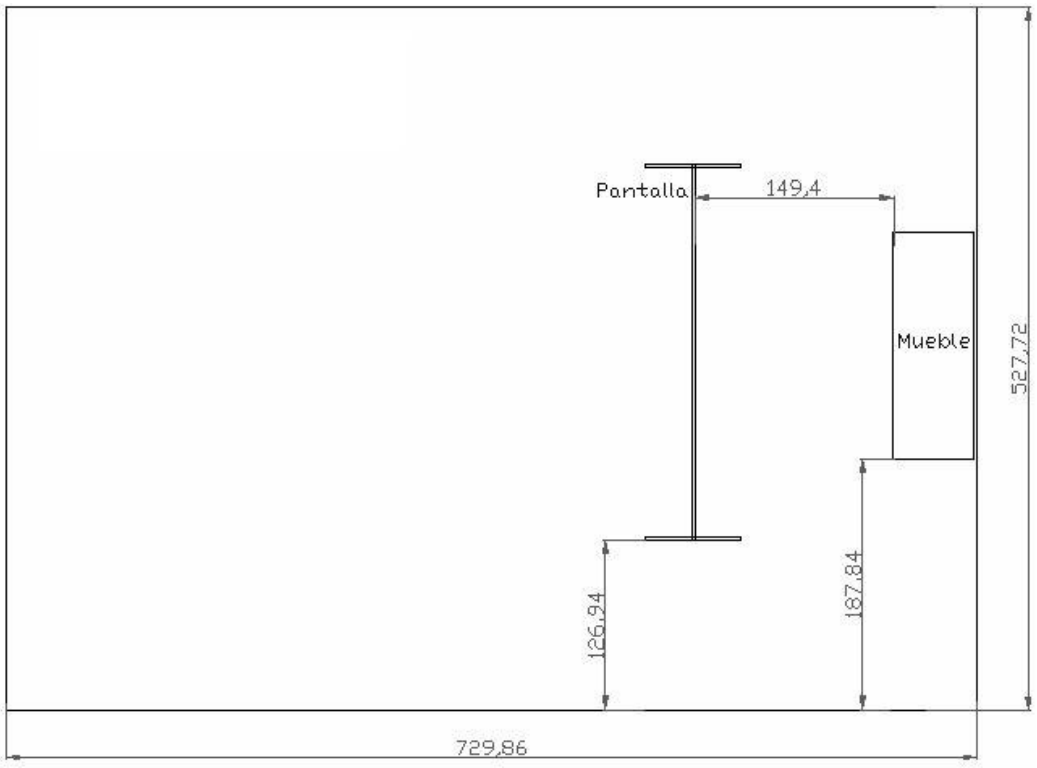


Figura 3-13 Sala del sistema de visualización

CAPÍTULO 4

4 IMPLEMENTACIÓN

En este capítulo se hace una descripción de los pasos seguidos para la instalación y configuración del *cluster* de computadores. Además, se especifican los pasos seguidos para la instalación y configuración del sistema distribuido de renderización. También se describe el ensamblaje y alineación de la pantalla y los proyectores del sistema de proyección.

4.1 Ensamblaje, instalación y configuración del *cluster*

Para la construcción del *cluster* se utilizaron cuatro computadores genéricos (Figura 4-1) con las siguientes características:

- procesador Intel Pentium 4 de 1.8 GHz,
- memoria RAM de 256 Mb,
- disco duro de 40 Gb,
- tarjeta de vídeo Biostar SiS 300 con 32 Mb de memoria (Figura 4-2), y,
- tarjeta de red de 10/100 Mbps.

Figura 4-1 Computadores para el *cluster*

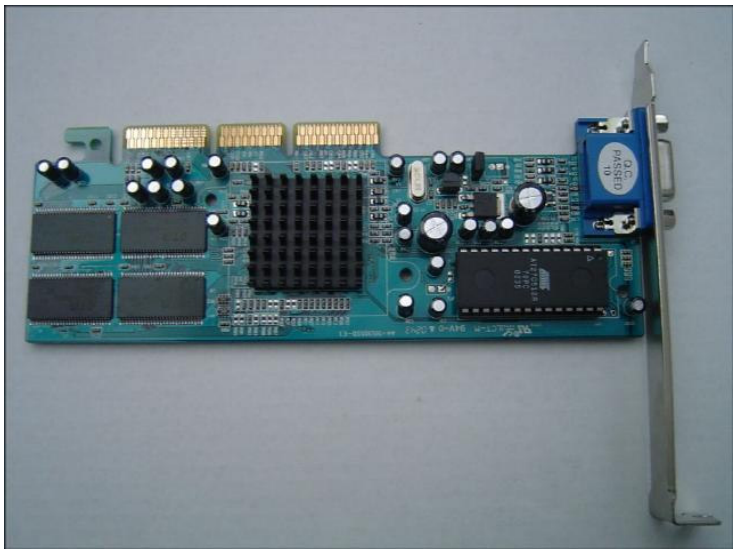


Figura 4-2 Tarjeta de vídeo Sis 300

Para la comunicación entre los computadores se usó un *switch* Ethernet de 10/100 Mbps marca D-Link modelo DES-1008D (Figura 4-3).

Figura 4-3 *Switch* Ethernet de 10/100 Mbps

El sistema operativo que se instaló en los computadores fue *Clic Fase 1* basado en la versión 9.0 de Mandrake. Cuando comienza la instalación se escoge el modo experto (Figura 4-4). Luego se escoge el ratón y el idioma del teclado. En el nivel de seguridad se pone *Standard*, para evitar la instalación de *firewalls* que pueden dificultar la comunicación entre los computadores del *cluster*.

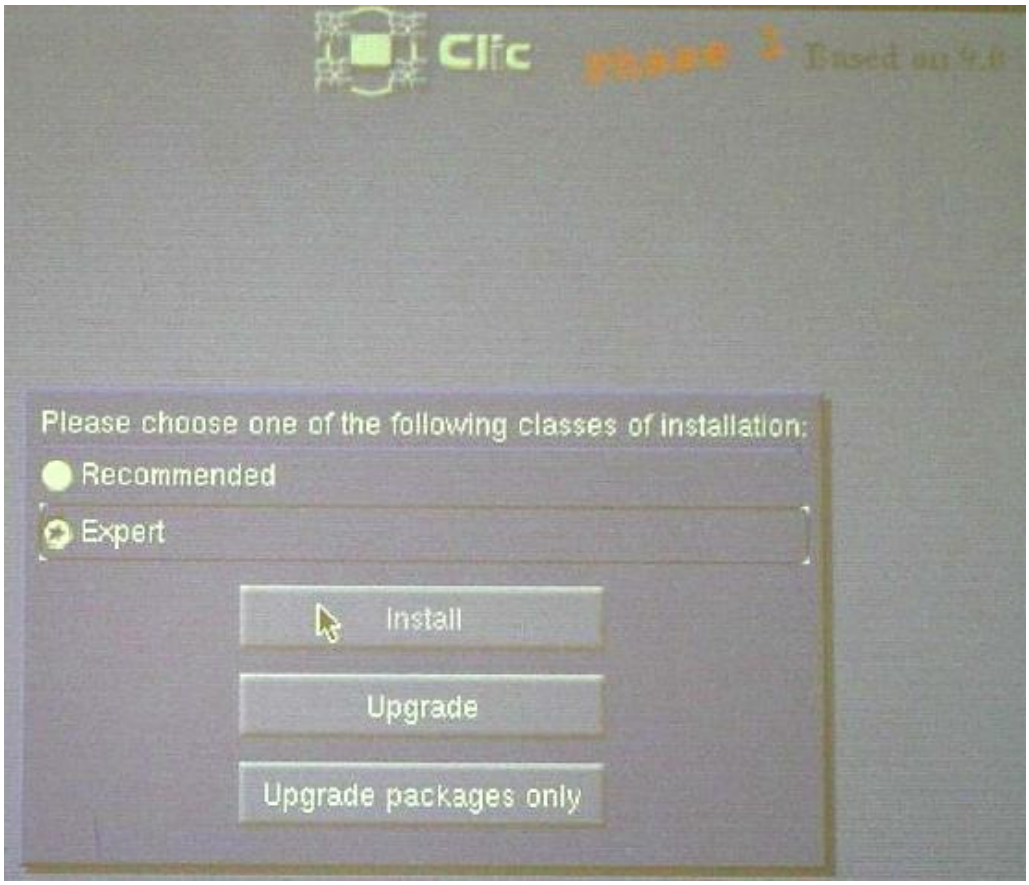


Figura 4-4 Instalación de Mandrake Clic

Se realizaron dos particiones en los discos duros de los computadores. La primera partición, de 10 Gb, se la dejó para uso exclusivo del sistema operativo (Figura 4-5). El resto del disco duro se lo dejo para los datos, utilizando como punto de montaje el directorio /home.

Al final las particiones del disco duro quedaron como se muestra en la Figura 4-6.

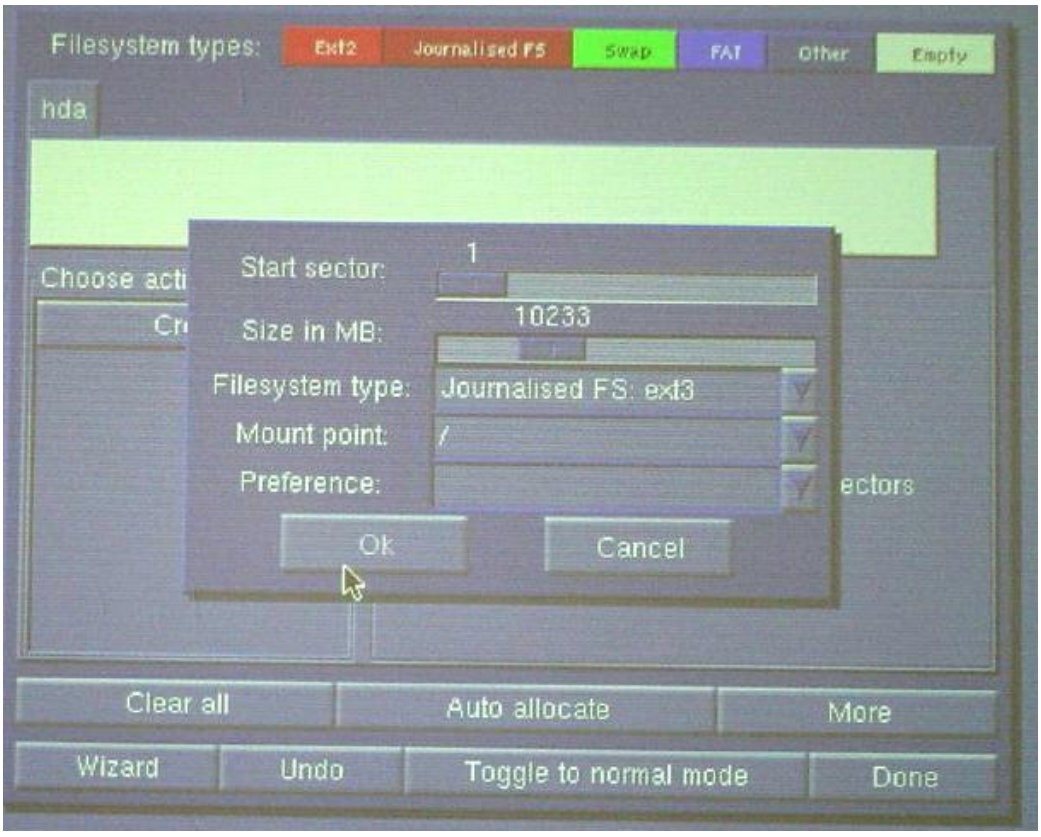


Figura 4-5 Partición para el sistema operativo

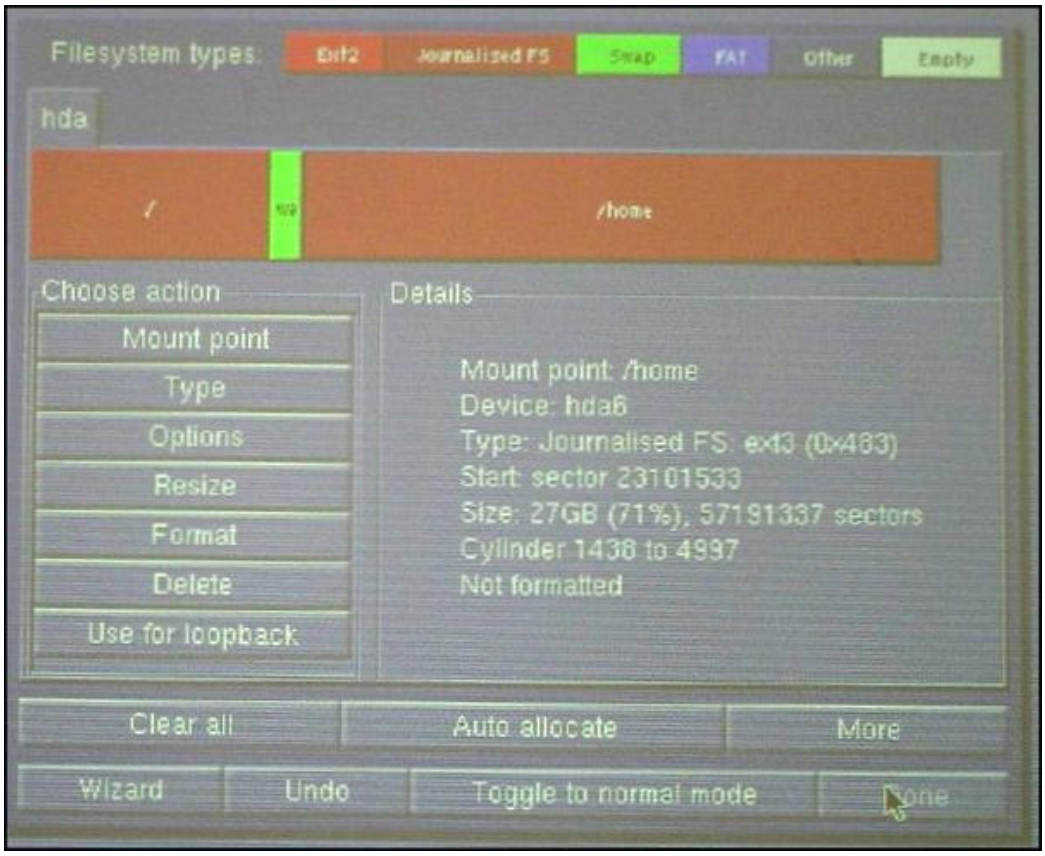


Figura 4-6 Partición del disco duro

Para la instalación solo se utilizaron las herramientas de *cluster*, ya que no se cuenta con las comerciales (Figura 4-7).

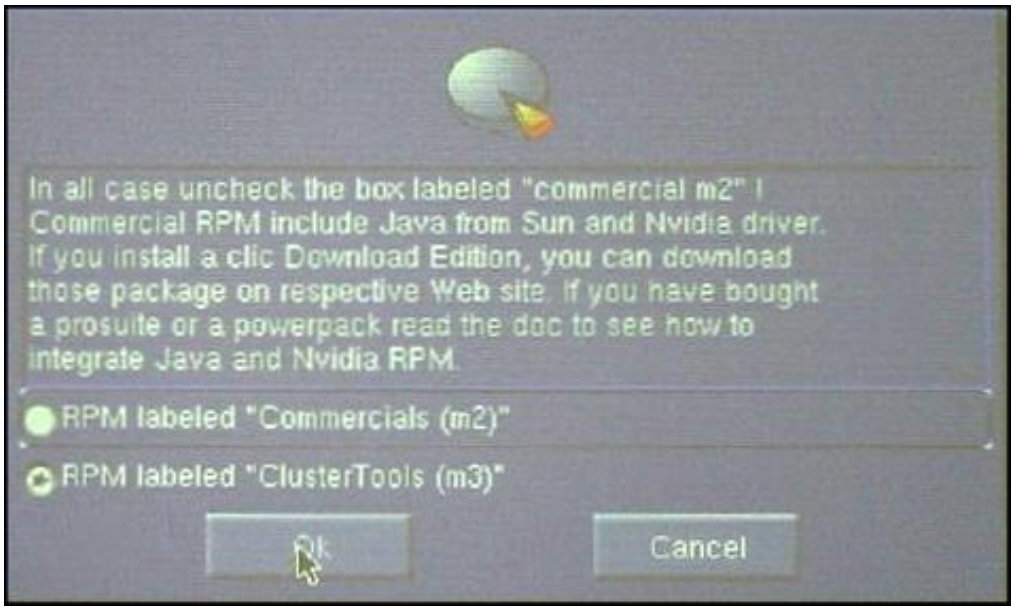


Figura 4-7 Opciones de instalación del *cluster*

En el servidor del *cluster* se instalaron los siguientes paquetes (Figura 4-8):

- *Cluster Benchmarking Tools*: Herramientas que miden el rendimiento del *cluster*.
- *Scientific Applications and Libraries*: Librerías y aplicaciones relacionadas a las ciencias y matemáticas.
- *Development*: Librerías necesarias para poder compilar nuevos programas.
- *Documentation*: Información de soporte del *cluster*.
- *Classical Cluster Server*: Librerías y herramientas necesarias para el funcionamiento del servidor del *cluster*.

- *Autoinstall Server*: Herramientas necesarias para la configuración automática del servidor.
- *DNS / NIS*: Herramientas necesarias para la configuración del servidor de dominio y el sistema de archivos compartido.
- *Basic System Configuration Tools*: Herramientas necesarias para la configuración del sistema.
- *Other Tools*: Para poder escoger herramientas y librerías adicionales.
- *Graphical Desktops*: Interfaces gráficas para la interacción con el usuario.

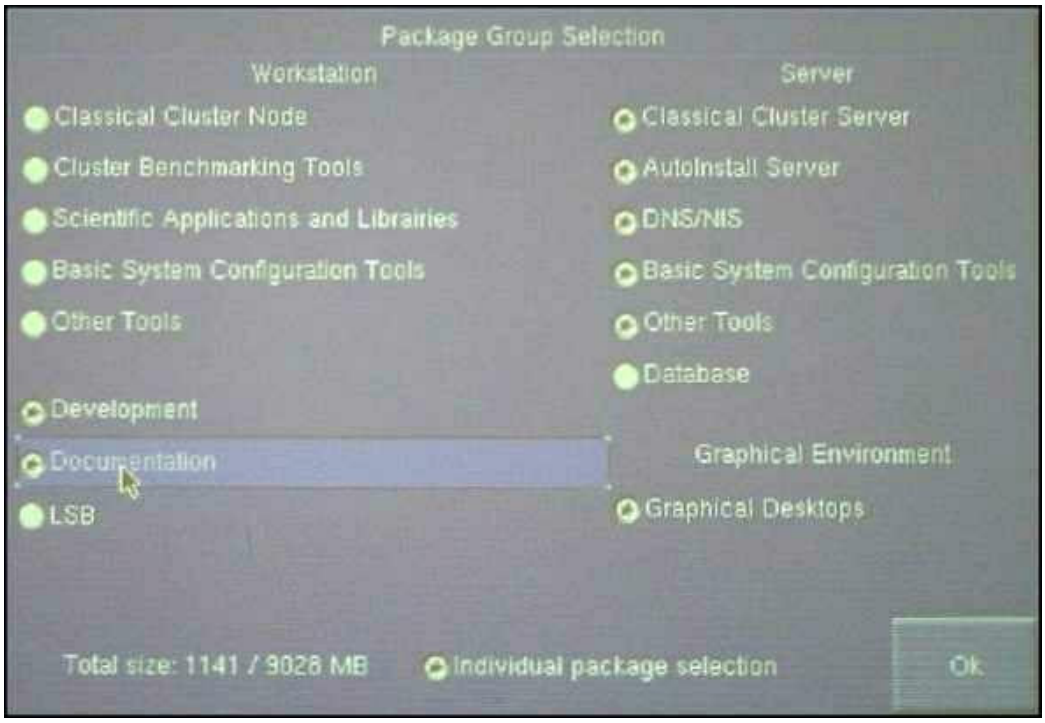


Figura 4-8 Paquetes instalados en el servidor

De los paquetes individuales solamente se escogió *Mesa*, que se encuentre en *Graphical Environment, Graphical Desktops, Other*. Este paquete instala la librería *libMesaGL*, como se puede ver en la Figura 4-9. Esta librería es la encargada de ejecutar los comandos OpenGL en Linux.

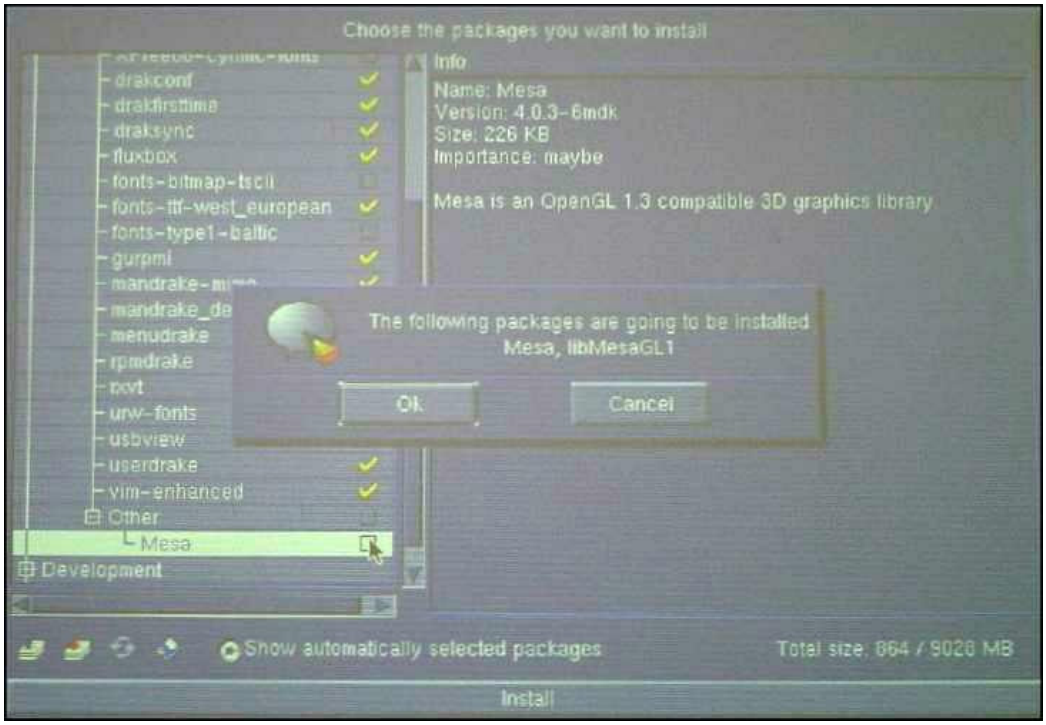


Figura 4-9 Instalación de Mesa

En los nodos, en cambio, se instalaron los siguientes paquetes:

- *Classical Cluster Node*: Librerías y herramientas necesarias para el funcionamiento de un nodo del *cluster*.

- *Cluster Benchmarking Tools*
- *Scientific Applications and Libraries*
- *Basic System Configuration Tools*
- *Other Tools*
- *Graphical Desktop*s

De la misma forma que con el servidor, se incluyeron los paquetes individuales de *Mesa* y la librería *libMesaGL*.

La instalación del sistema operativo tomó aproximadamente 7 minutos por computador (Figura 4-10).

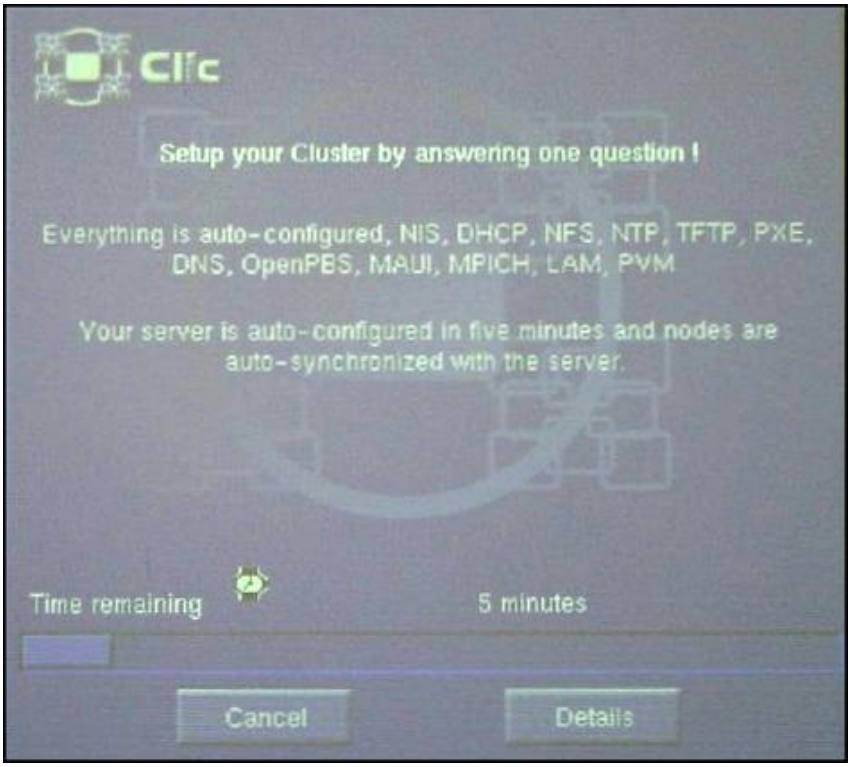


Figura 4-10 Tiempo de instalación

Después de terminar la instalación de los paquetes, se determina la contraseña del usuario *root*, y luego se configura la red. Al servidor se le asignó la dirección 200.10.150.140, los nodos en cambio deben tener habilitada la opción *Automatic IP* (DHCP) (Figura 4-11). Como máscara de subred se utilizó 255.255.255.0. En la siguiente pantalla se introdujo el *hostname*, para el servidor es node140.cti.espol.edu.ec. Los nodos en cambio deben dejar en blanco esta opción.

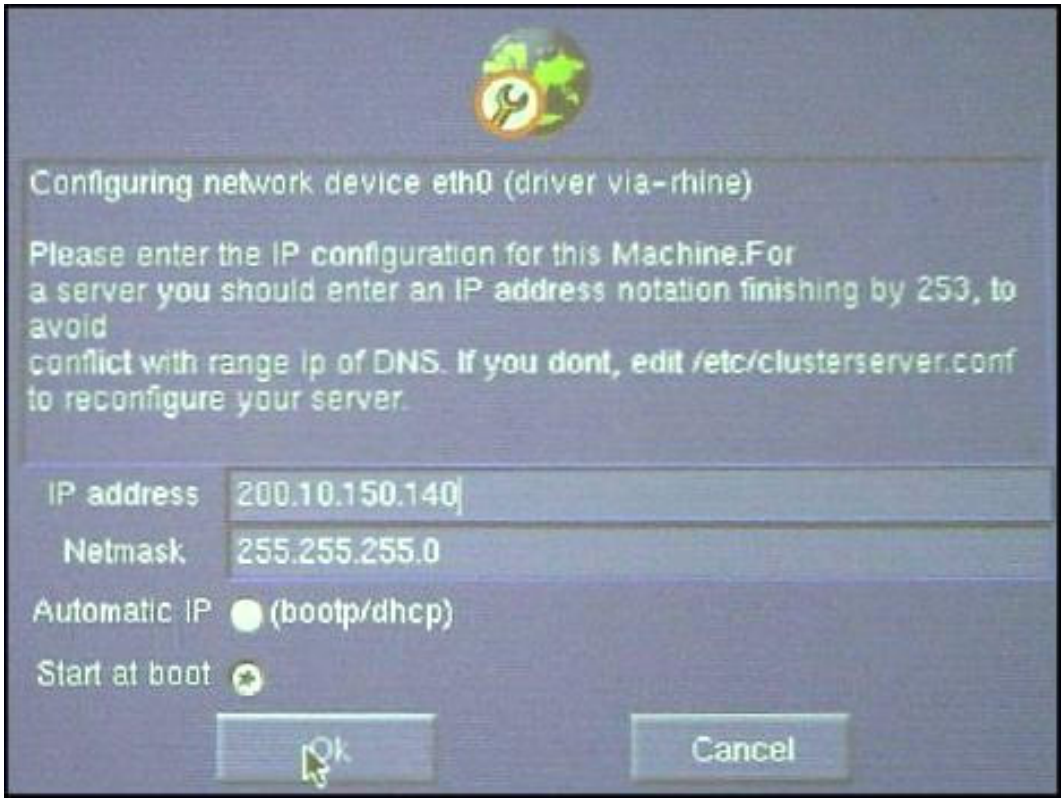


Figura 4-11 Configuración de la red

Las opciones de arranque se las deja con las mismas que vienen por defecto. El monitor se lo configuró como genérico de 800 x 600 a 60 Hz.

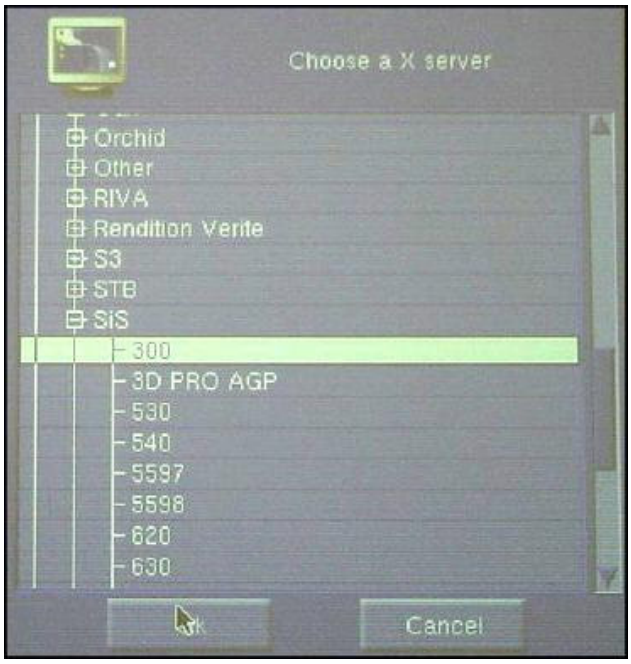


Figura 4-12 Configuración de la tarjeta de vídeo

Por último, antes de finalizar la instalación, se configuró la tarjeta de vídeo. Para el X Server se escogieron los *drivers* de la tarjeta gráfica SiS 300 (Figura 4-12). Se seleccionó 800 x 600 a 16 bits para la resolución, ya que esa es la resolución nativa de los proyectores (Figura 4-13).

Además se escoge que empiece con la interfaz gráfica por defecto cada vez que arranca el computador.

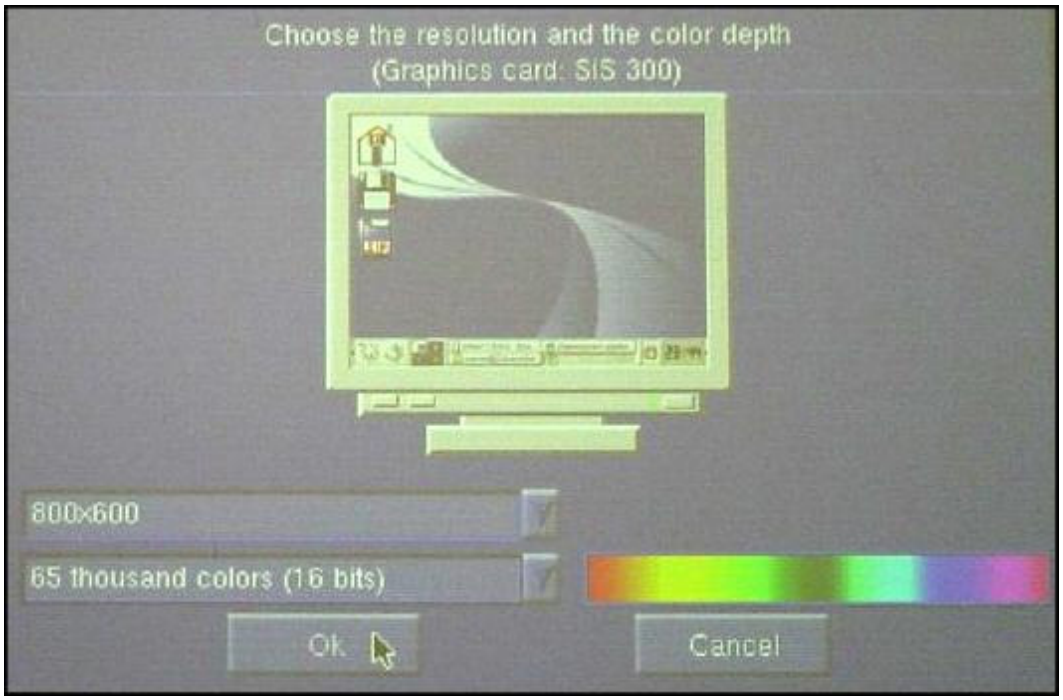


Figura 4-13 Resolución de los monitores

Una vez terminada la instalación de Mandrake Clic tanto en el servidor como en los nodos, se procedió a la configuración del *cluster*.

Como se indicó en la sección 2.2.3, Clic provee las herramientas para que la configuración se realice automáticamente. Antes de iniciar la configuración hay que cambiar el archivo *clusterserver.conf* que se encuentra en el directorio /etc/. El rango de direcciones de los nodos que se encuentra en ese archivo no debe incluir la dirección del servidor. Al final el archivo debe quedar como se muestra en la sección A.1 del Apéndice A.

Luego hay que insertar el disco de instalación y entrar como *root* al servidor.

Si se tienen problemas con la lectura del CD, se digita:

```
eject; eject -t;
```

Posteriormente en una ventana de comandos se digita la siguiente sentencia:

```
setup_auto_server
```

Esto copia el contenido del CD en el servidor y comienza con la configuración de los distintos servicios.

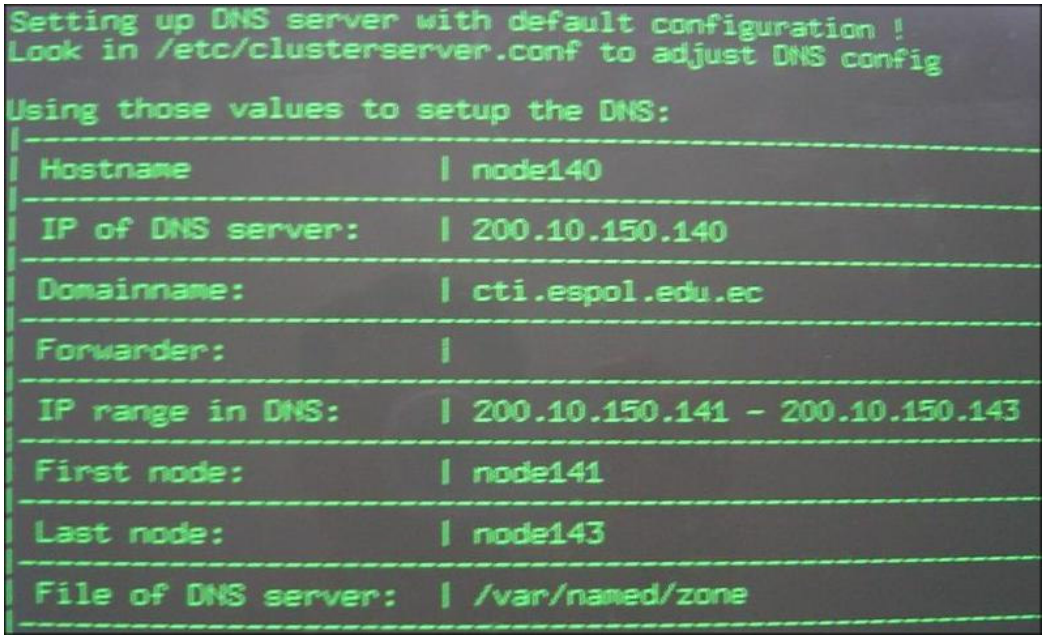


Figura 4-14 Configuración del DNS

Primero se configura la red, y se pregunta cuantos nodos tendrá el *cluster* (en el caso de este sistema 3). Luego se configuran automáticamente el DNS (Figura 4-14), el NIS y el DHCP. Por último se recopilan las direcciones MAC de las tarjetas de cada nodo para asociarlas con una dirección IP (Figura 4-15). En este momento se deben encender los nodos restantes del *cluster* para que sean detectados.

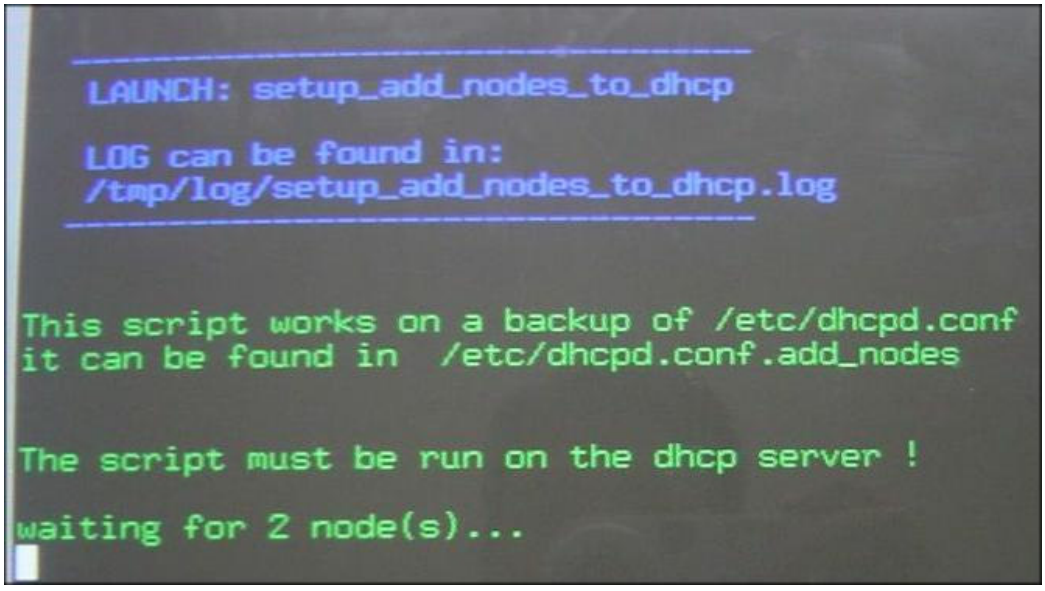


Figura 4-15 Asociación de las direcciones MAC de los nodos a su IP

Una vez concluida la configuración automática, el *cluster* está listo para ser utilizado.

Cabe indicar que para este sistema el escritorio que se utilizó como interfaz es WindowMaker, el cual fue instalado con el Mandrake Clic (Figura 4-16).

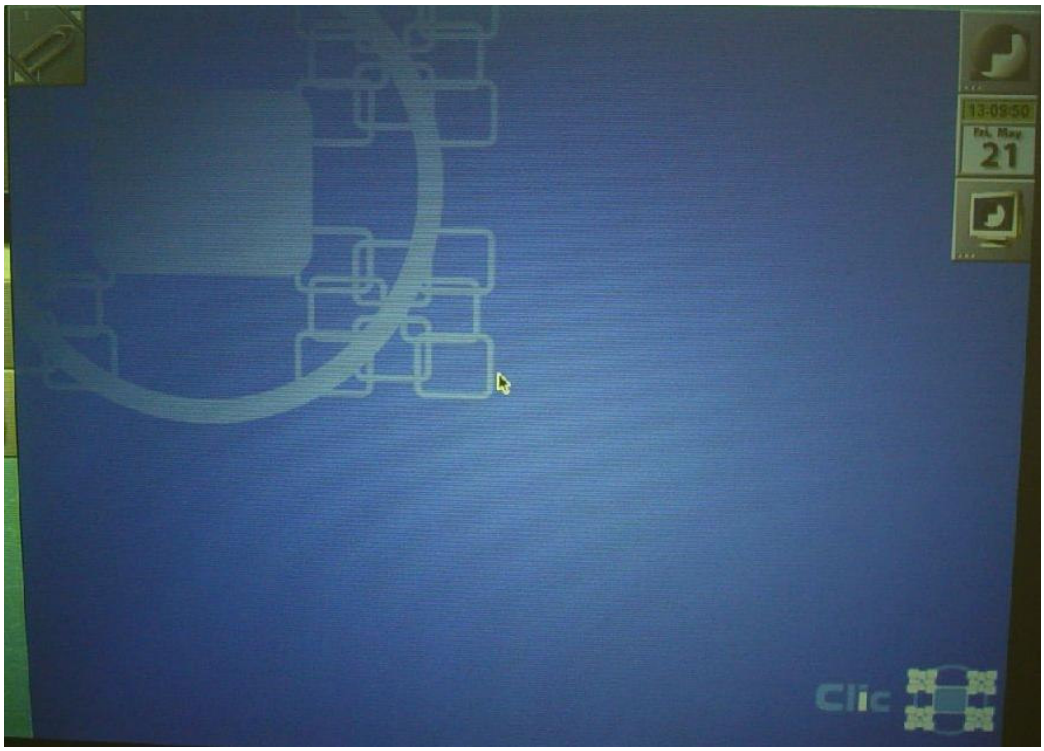


Figura 4-16 Pantalla principal de WindowMaker

Para iniciar la sesión en el *cluster* se puede utilizar el usuario *root*, pero por cuestiones de seguridad es preferible crear un usuario que no tenga todos los permisos. Además este nuevo usuario poseerá una carpeta que será compartida por todos los computadores del *cluster*, por lo que esta carpeta puede ser utilizada tanto para los archivos de

configuración, como para la instalación de los programas que se necesitan correr en todas las máquinas.

Para crear un nuevo usuario hay que ir a la administración de usuarios de Clic y presionar en *adduserNis* (Figura 4-17).

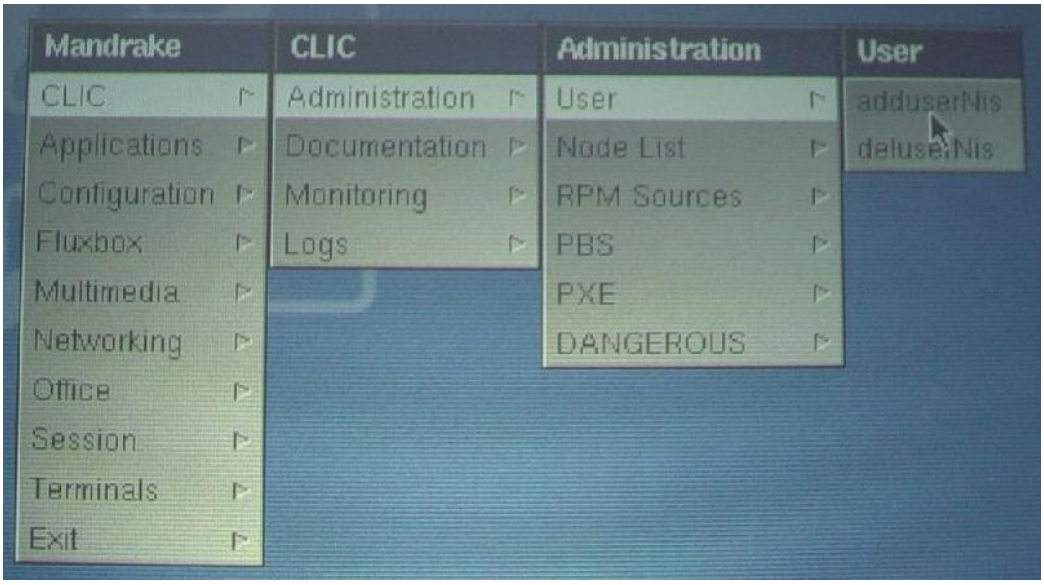


Figura 4-17 Agregar usuario

Una vez ingresado el nombre del nuevo usuario (en el caso de este sistema se creó un usuario llamado “mural”), este se lo agrega a los grupos que uno desee y se ingresa un comentario sobre quien es el usuario (Figura 4-18).

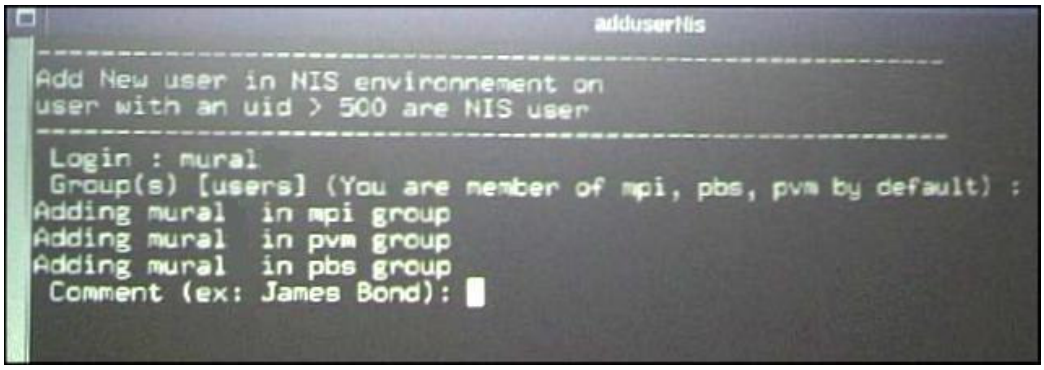


Figura 4-18 Creación de usuario mural

Para facilitar el uso del *cluster*, los nodos deben ser configurados de tal modo que el inicio de la sesión (*log in*) sea automático. Para esto hay que ingresar al *Mandrake Control Center* (Figura 4-19).

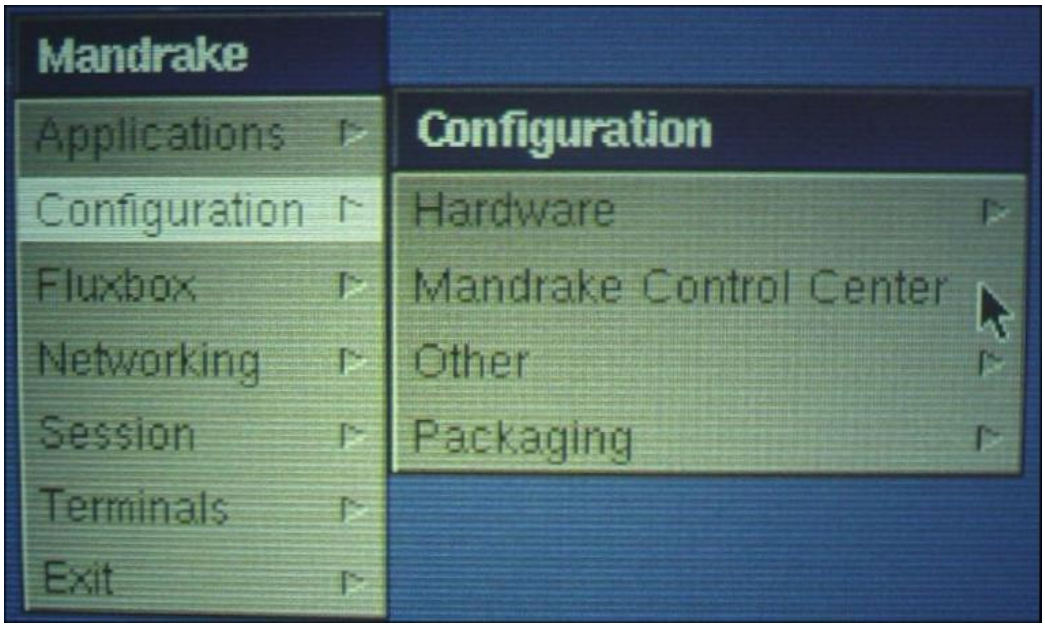


Figura 4-19 Mandrake Control Center

Dentro del *Mandrake Control Center* hay que ir a la opción *Boot* y luego a *Boot Config* (Figura 4-20).

Dentro de esta configuración hay que escoger la opción "*Yes, I want autologin with this (user, desktop).*" En el campo de usuario se pone "mural" y en el del escritorio se pone "WindowMaker".

Figura 4-20 Configuración de inicio

Además se debe instalar en todos los computadores el *XFree 4*. Para esto se ingresa nuevamente al *Mandrake Control Center* (Figura 4-19), se escoge la opción *Software Management*, y luego *Install Software* (Figura 4-21).

Figura 4-21 Instalación de software

Se buscan todos los paquetes que tengan “XFree”, y se instalan los siguientes:

```
XFree86-doc-4.2.1-3mdk
XFree86-server-4.2.1-3mdk
```

Luego se desinstala el XFree 3. Para esto se ingresa a la opción *Remove Software* en el *Mandrake Control Center* (Figura 4-22).

Figura 4-22 Desinstalación de software

Se buscan todos los paquetes que tengan “XFree”, y se desinstalan los siguientes:

```
XFree86-server-common-3.3.6-28mdk
XFree86-SVGA-3.3.6-28mdk
```

Después se borra el vínculo viejo del X, y se establece el nuevo digitando:

```
rm /etc/X11/X
ln -s /usr/X11R6/bin/XFree86 /etc/X11/X
```

Por último los archivos de configuración *XF86Config* y *XF86Config-4*, que se encuentran en /etc/X11/ debe quedar tal como se muestra en la secciones A.2 y A.3 del Apéndice A.

Además el archivo *.xinitrc*, que se encuentra en /home/nis/mural/ y determina lo que deben hacer los computadores en caso de no encontrar el WindowMaker, debe quedar como se muestra en la sección A.4 del Apéndice A.

4.2 Instalación y configuración del sistema distribuido de renderización

Para la distribución del escritorio se utilizó el DMX Snapshot de marzo de 2003. Para descomprimir el archivo, en el computador que actúa como servidor, se pone la sentencia:

```
bunzip2 dmx-snapshot-20030331.tar.bz2
```

Luego se debe volver a descomprimir de la siguiente manera:

```
tar -xvf  dmx-snapshot-20030331.tar
```

Una vez que el archivo de instalación está descomprimido, hay que compilarlo. Para esto, hay que copiar el archivo *host.def-optimize* que se encuentra en /home/nis/mural/dmx/doc/ al directorio /home/nis/mural/dmx/xc/config/cf/. Se le cambia el nombre a *host.def* y se lo edita de tal forma que quede como se muestra en la sección A.5

del Apéndice A. Luego dentro del directorio /home/nis/mural/dmx/xc/ se lo compila digitando la sentencia:

```
make World
```

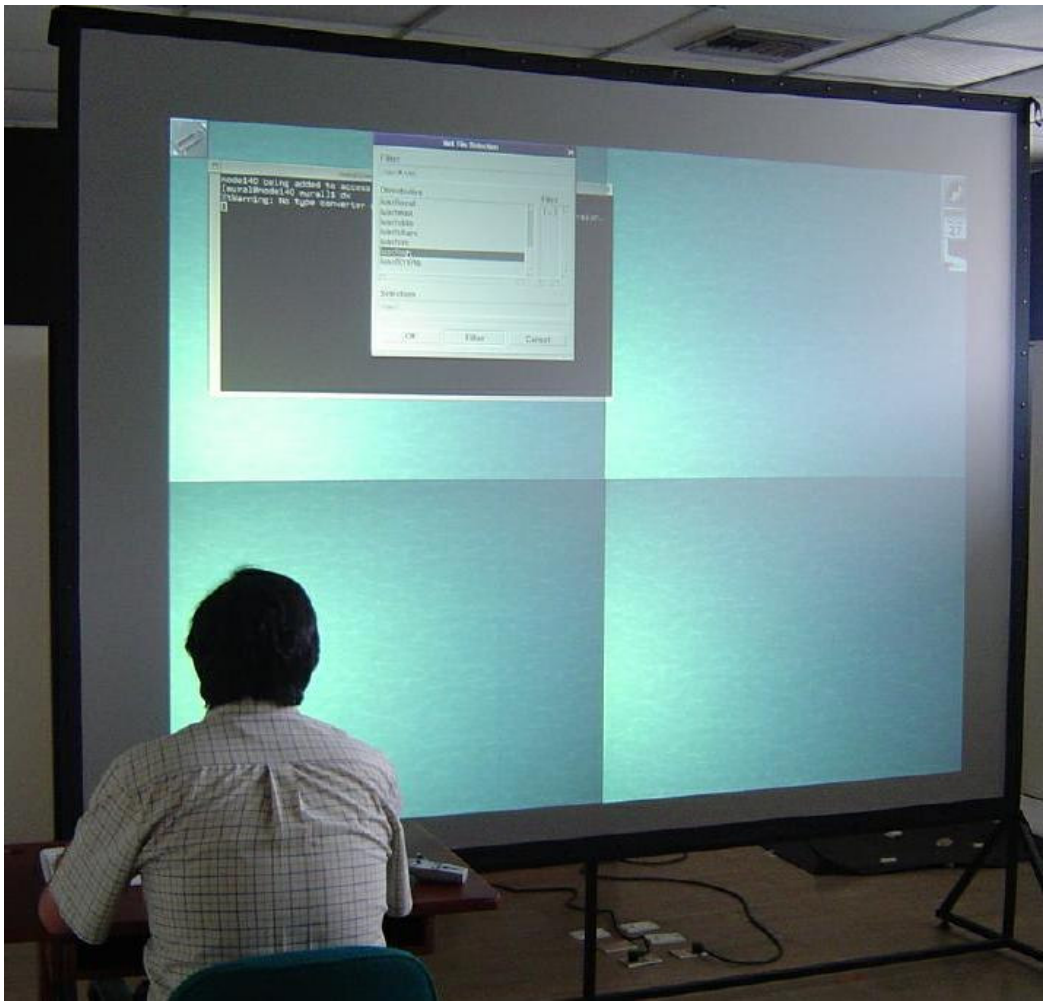


Figura 4-23 Escritorio distribuido por DMX

El proceso de compilar DMX puede tomar varias horas. Una vez concluido, se debe configurar DMX. Lo único que hay que hacer es

crear un archivo con el nombre *dmx.conf* en el directorio /home/nis/mural/dmx/, en este archivo se indica que parte de la pantalla mostrará cada computador, y debe quedar como se muestra en la sección A.6 del Apéndice A. Además se debe dar permiso al servidor para manipular los nodos. Para esto se edita el archivo *.bashrc* que se encuentra en el directorio /home/nis/mural/ y se agrega la línea:

```
xhost +node140
```

Para probar que DMX si está funcionando, en una ventana de terminal del servidor se digita:

```
startx -- /home/nis/mural/dmx/xc/programs/Xserver/Xdmx :1
+xinerama -configfile /home/nis/mural/dmx/dmx.conf -config
test
```

Esta sentencia señala donde se encuentran el archivo ejecutable y el archivo de configuración. Además indica que se utilizará *xinerama*.

Una vez digitada la sentencia el escritorio se repartirá entre los cuatro computadores, tal como se ve en la Figura 4-23.

Para que cada computador vuelva a mostrar su propio escritorio se presiona *Ctrl-Alt-Q* o se cierra la sesión del usuario actual.

Una vez lograda la distribución del escritorio, se debe instalar Chromium para distribuir las operaciones gráficas, pero para que

funcione Chromium, es necesario antes las siguientes librerías: Python, LibMagick, y GLUT.

Para instalar Python (versión 2.4) se digita:

```
rpm -ivh wxPythonGTK-py2.2-2.4.1.2-1.i386.rpm
```

Se utilizó la versión 5.5.1 de LibMagick. Pero esta requiere que la librería liblcms.so.1 sea instalada previamente, por lo que primero se digita:

```
rpm -ivh liblcms1-1.10-1mdk.i586.rpm
```

Luego se instala LibMagick digitando:

```
rpm -ivh libMagick5.5.1-5.5.1.6-1mdk.i586
```

Por último se instaló la versión 3.7 de GLUT. Primero se lo descomprime digitando:

```
tar -zxvf glut-3.7.tar.gz
```

Luego hay que compilarlo, pero para esto primero se copia el archivo *Glut.cf* que está en /glut-3.7/linux/ en el directorio /glut-3.7/. Luego en el directorio glut-3.7 se digita:

```
./mkmkfiles.imake
```

Después se copia el archivo *Makefile* que se encuentra en /glut-3.7/linux/ al directorio /glut-3.7/lib/glut/. Estando en este último directorio se digita *make* para compilar la librería.

Luego se copia el archivo *libglut.so.3.7* en los directorios /usr/X11R6/lib/ y /usr/lib/. Además se copia el archivo *glut.h* que se encuentra en /glut-3.7/include/GL/ en la carpeta /usr/X11R6/include/GL/

Por último en las carpetas /usr/X11R6/lib/ y /usr/lib/ se crean los vínculos a la nueva librería digitando:

```
ln –s libglut.so.3.7 libglut.so.3
ln –s libglut.so.3.7 libglut.so
```

Una vez que han sido instaladas todas estas librerías, se deben copiar unas librerías de DMX para que Chromium pueda correr cuando se utiliza DMX. Para esto se digitan las siguientes sentencias:

```
cp /home/nis/mural/dmx/xc/include/extensions/dmxext.h
/usr/X11R6/include/X11/extensions
cp /home/nis/mural/dmx/xc/exports/lib/libdmx.a
/usr/X11R6/lib
```

Al concluir esto, se puede proceder a la instalación de Chromium. Se utilizó la versión 1.2 de este programa.

Para descomprimir el archivo se digita:

```
tar –zxvf cr-12tar.gz
```

Luego se cambia el archivo *Linux.mk* que se encuentre dentro de la carpeta config, y en *USE_DMX* se pone 1.

Para compilarlo se digita *"make"* en la carpeta de Chromium. Una vez compilado, en todos los computadores se copian todas las librerías que están en /home/nis/mural/cr-1.2/lib/Linux/ a /usr/lib/.

Por último, en el directorio /home/nis/mural/cr-1.2/lib/Linux/ se establecen los vínculos para que las llamadas a OpenGL sean interceptadas por Chromium digitando:

```
ln –s libcrfaker.so libGL.so
ln –s libcrfaker.so libGL.so.1
```

Para comprobar que Chromium si esté funcionando se necesitan abrir tres consolas. En la primera se digitan las siguientes líneas:

```
cd /home/nis/mural/cr-1.2/mothership/configs/
export CRMOTHERSHIP=node140
python crdemo.conf atlantis
```

Esto indica al computador que el *node140* es el servidor, y además que se va a utilizar la configuración que está en el archivo *crdemo.conf* para correr el programa *atlantis*.

En otra ventana se digita:

```
export CRMOTHERSHIP=node140
```

```
export PATH=$PATH: /home/nis/mural/cr-1.2/bin/Linux/
export LD_LIBRARY_PATH=/home/nis/mural/cr-1.2/lib/Linux/
crserver
```

Estas sentencias establecen las variables del sistema, indicando el nodo que actúa como servidor, el lugar donde se encuentran los archivos ejecutables de Chromium, y el lugar donde se encuentran las librerías. Por último hace que el servidor corra.

Por último en la tercera se pone:

```
export CRMOTHERSHIP=node140
export PATH=$PATH: /home/nis/mural/cr-1.2/bin/Linux/
export LD_LIBRARY_PATH=/home/nis/mural/cr-1.2/lib/Linux/
crappfaker
```

Estas sentencias establecen las variables del sistema, y al final corre un programa de prueba en el cual salen varios delfines nadando por la pantalla como se puede ver en la Figura 4-24.

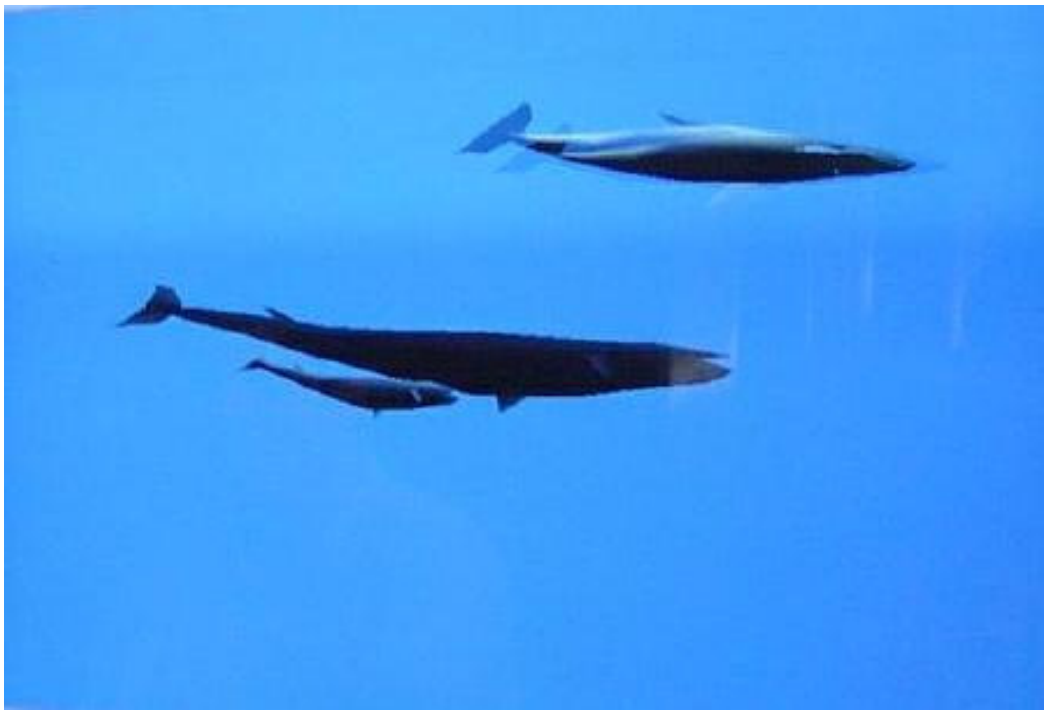


Figura 4-24 Atlantis

Para automatizar la utilización de Chromium, los archivos *.bashrc*, *.crconfigs* y *.crsite* que se encuentran en /home/nis/mural/ deben quedar tal como se muestran en las secciones A.7, A.8 y A.9 del Apéndice A, respectivamente.

En el archivo *.bashrc*, se establece la ruta donde se encuentran las librerías de Chromium y se le da al servidor permisos para acceder cada nodo.

En el archivo *.crconfigs* se indica la ruta donde se encuentra el archivo de configuración que Chromium debe utilizar.

En el archivo *.crsite* se determina el tamaño de la pantalla y los nodos que la conforman.

Por último se debe crear un archivo de configuración para que cada vez que corra DMX, Chromium también corra y esté listo para interceptar las llamadas OpenGL. Este archivo se llama *miautodmx.conf* y debe estar en el directorio /home/nis/mural/cr-1.2/mothership/configs/.

En este archivo se indica donde está el directorio que contiene a Chromium (en este caso /home/nis/mural/cr-1.2/), la cantidad de filas y columnas que conforman la pantalla, y las instrucciones que se deben ejecutar en cada nodo para que Chromium pueda distribuir las operaciones gráficas. Al realizar los cambios correspondientes a este sistema, el archivo queda tal como se muestra en la sección A.10 del Apéndice A.

Luego de que se han instalado todas las librerías, Chromium y se han realizado los cambios en los archivos de configuración anteriormente mencionados, cada vez que, utilizando DMX, se corra un programa que utilice OpenGL, automáticamente Chromium interceptará estas llamadas y distribuirá las operaciones gráficas. Sin embargo, esta distribución será totalmente transparente para el usuario, el cual podrá correr el programa como lo hace normalmente en un computador.

4.3 Ensamblaje del sistema de proyección

El ensamblaje del sistema de proyección consiste básicamente en dos actividades: armar la pantalla, y, colocar y alinear los proyectores en el mueble.

Se utilizó una pantalla de proyección trasera modelo *Cineperm* de marca *Draper*. Su tamaño es 203 x 274 cm., y su área de proyección interna es 193 x 264 cm. El área reproyección de la pantalla es de un material llamado *Cineplex*, que consiste en vinilo flexible de color gris, además tiene un filo de dos pulgadas de cuero artificial que se adhiere a un marco de aluminio de una pulgada.

El montaje de la pantalla fue bastante sencillo. El marco de la pantalla posee 66 botones que no solamente sostienen la pantalla, sino que también la templan. En las Figura 4-25, Figura 4-26 y Figura 4-27 se puede ver el proceso de armado de la pantalla.

Figura 4-25 Montaje de la pantalla de proyección

Figura 4-26 Montaje de la pantalla de proyección

Figura 4-27 Montaje de la pantalla de proyección

Fue necesario construir una base para sostener la pantalla, cuyo diseño se puede ver en la Figura 3-12.

La pantalla fue unida con la base a través de tornillos (Figura 4-28). Y finalmente la pantalla quedó como se muestra en la Figura 4-29.

Figura 4-28 Unión de la pantalla con la base

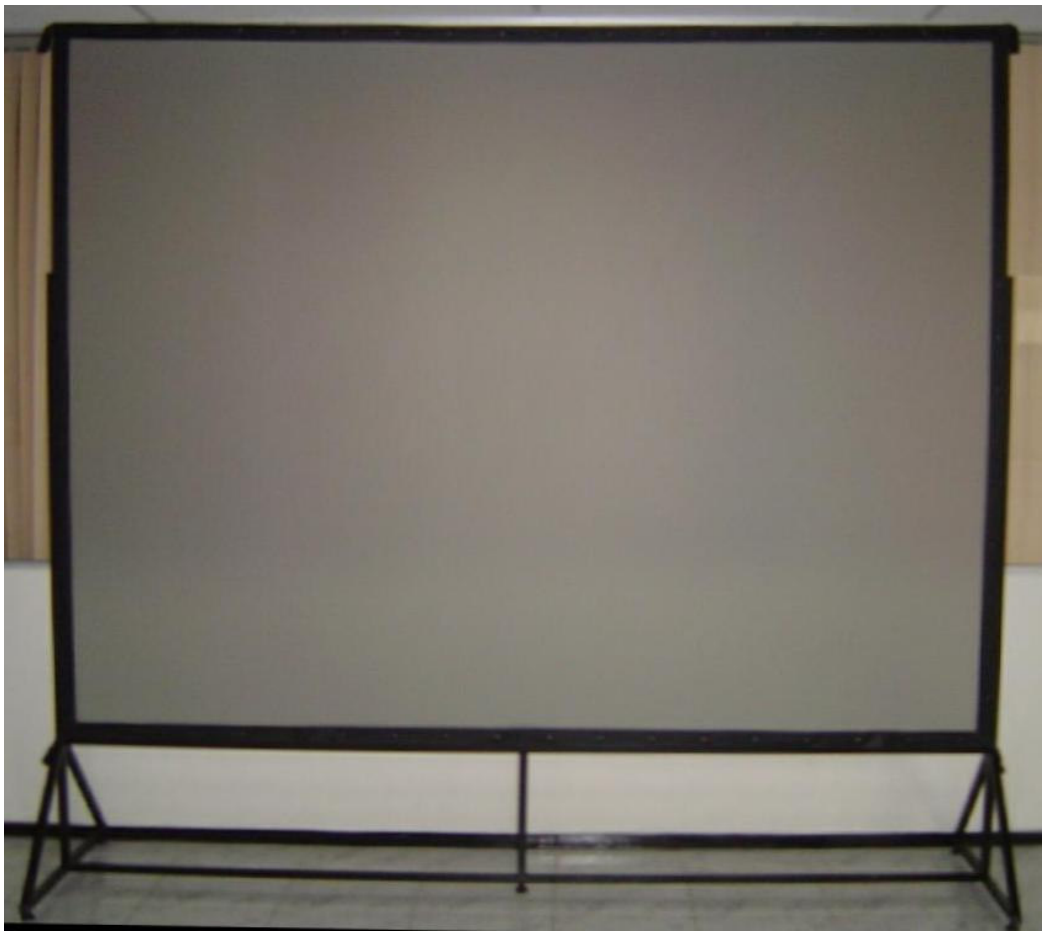


Figura 4-29 Pantalla de proyección trasera Draper Cineperm con su base

El mueble para los proyectores fue construido con el diseño que se muestra en la Figura 3-9, y se lo puede ver en la Figura 4-30.

En cada compartimento se colocó un computador y un proyector (Figura 4-31).

Figura 4-30 Mueble para los computadores y los proyectores

Figura 4-31 Computador y proyector en el mueble

Las bases de los proyectores (Figura 4-32) proveen los cuatro grados de libertad que se indicaron en la sección 3.3.1.

Figura 4-32 Base de un proyector

Para determinar la posición de cada proyector se utilizó una plomada en la pantalla (Figura 4-33). Sin embargo, como se puede ver en la Figura 4-34, no fue sencillo cuadrar los proyectores, porque a pesar de que las bases tenían cuatro grados de libertad, no era fácil mover los tornillos de precisión. Además, las bases no estaban alineadas entre sí, lo cual dificultó más la alineación de las proyecciones.

Además el piso del cuarto donde se encuentra la pantalla no era totalmente recto, lo cual dificultó aún más el proceso de posicionar los proyectores.

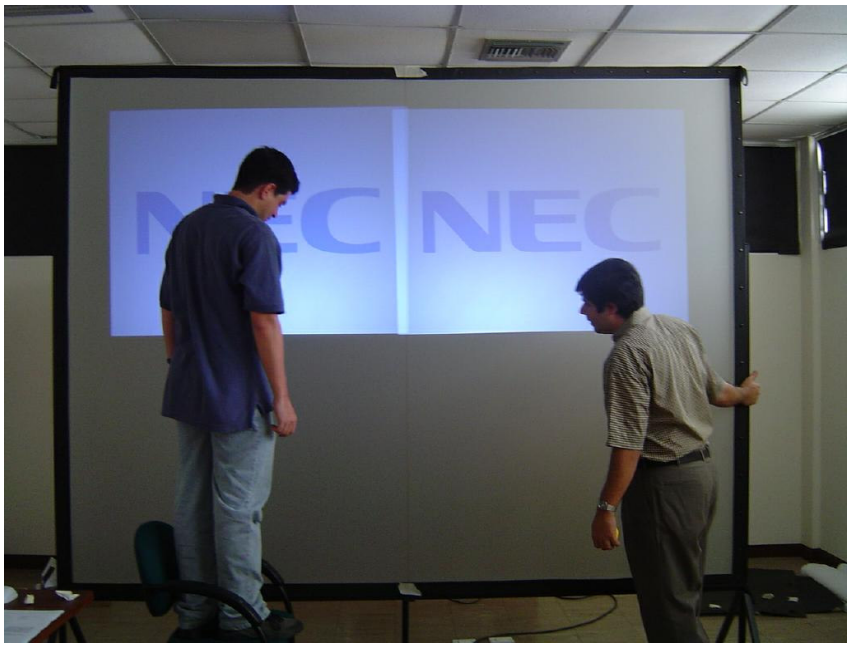


Figura 4-33 Plomada para cuadrar los proyectores

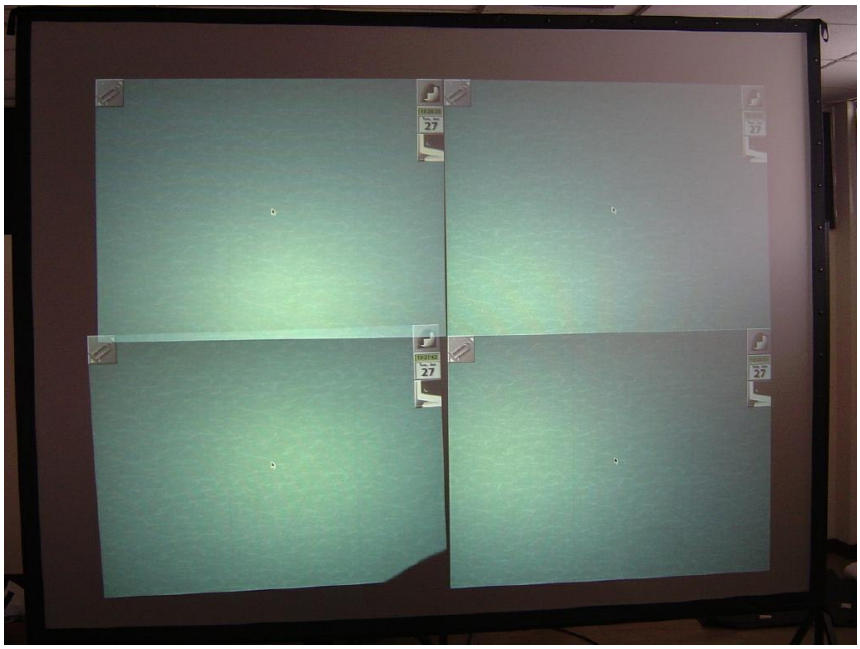


Figura 4-34 Proyectores sin alinear

Con la plomada se determinó la posición del proyector del servidor, es decir el que está arriba a la izquierda. Luego se alinearon los dos proyectores que tenían un lado común con este, es decir el otro de arriba y el de abajo a la izquierda. El tamaño de estas proyecciones se las fijó de acuerdo a la proyección del servidor. Por último se cuadró el proyector de abajo a la derecha, haciéndolo coincidir en el espacio que quedaba libre. Finalmente, después de muchas horas de trabajo, la pantalla quedó relativamente alineada. Como se puede ver en la Figura 4-35, quedaron pequeñas fallas, pero son casi imperceptibles.

Figura 4-35 Pantalla alineada

Se debe recalcar que la diferencia de brillo que se aprecia entre los proyectores depende del ángulo desde donde se ve la pantalla.

Una vez armada la pantalla y alineados proyectores, lo único que se cambió en el sistema fue la utilización de un ratón y un teclado inalámbrico (Figura 4-36) debido a que es muy difícil para el usuario utilizar la pantalla desde detrás, y con estos puede hacerlo desde el frente como se aprecia en la Figura 4-37.

Figura 4-36 Ratón y teclado inalámbrico

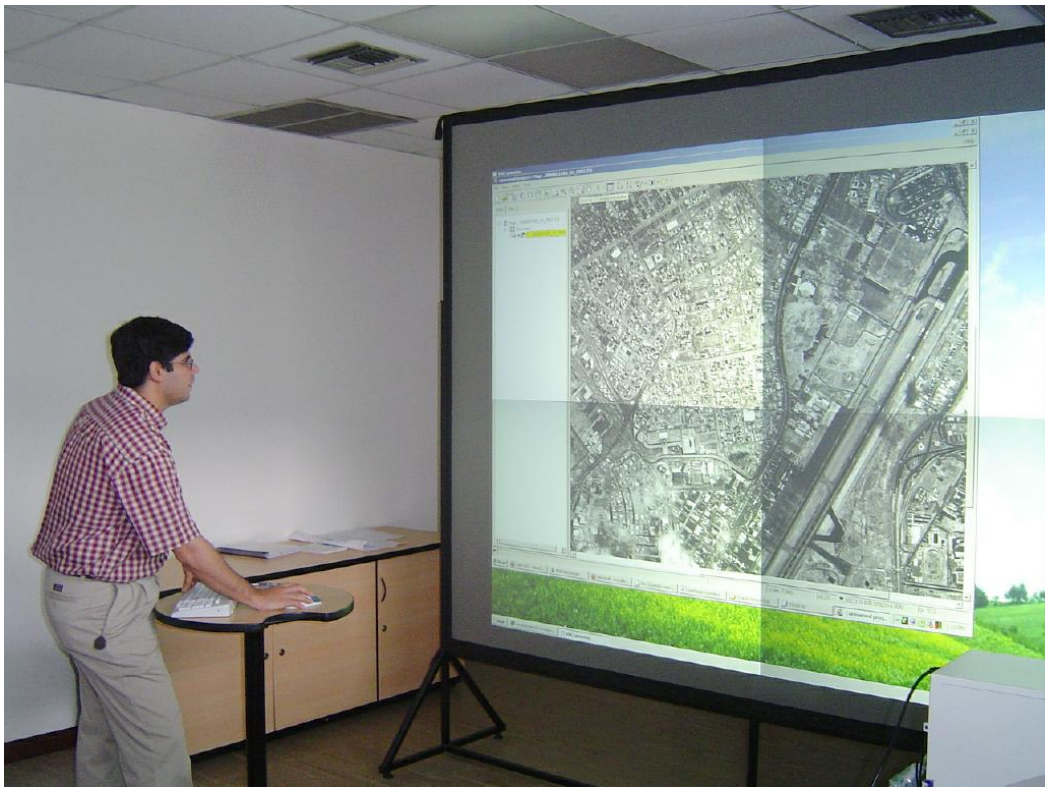


Figura 4-37 Usuario utilizando la pantalla desde el frente

CAPÍTULO 5

5 PRUEBAS DEL SISTEMA

En este capítulo se hace una descripción de las pruebas de funcionamiento del software de visualización (OpenDX y Vis5d+). De igual manera se hace una comparación del rendimiento del sistema con un sistema no distribuido y con otros sistemas distribuidos. Además, se muestran los resultados de pruebas del sistema en la colaboración y control remoto de aplicaciones. También se muestran los resultados de una prueba del sistema con datos reales obtenidos en la ESPOL.

5.1 Pruebas de funcionamiento

5.1.1 OpenDX

Se utilizó la versión 4.2 de OpenDX. Este programa necesita de la librería *libMagick* para funcionar, pero como esta fue instalada previamente para que funcione Chromium, no es necesario reinstalarla. Además se necesita la librería *lesstif*. Esta se encuentra en el disco de Mandrake Clic, y se la instala con el mismo procedimiento con el cual se instaló el XFree en la sección 4.1, es decir a través del *Mandrake Control Center*.

Una vez instaladas las librerías, se instala el programa digitando:

```
rpm -ivh --nodeps OpenDX-4.2.0-5mdk.i586.rpm
```

Además se instalaron dos paquetes de ejemplos para este programa. Para esto se digita:

```
rpm -ivh opendx-samples-420-1noarch
tar -zxvf dx_21_earth_spacetar.gz
```

El primer paquete se instala automáticamente en /usr/lib/dx/samples/. En cambio, el segundo se instala el directorio donde se encuentra el archivo de instalación, por lo tanto hay que copiarlo a /usr/lib/dx/samples/ antes de instalarlo.

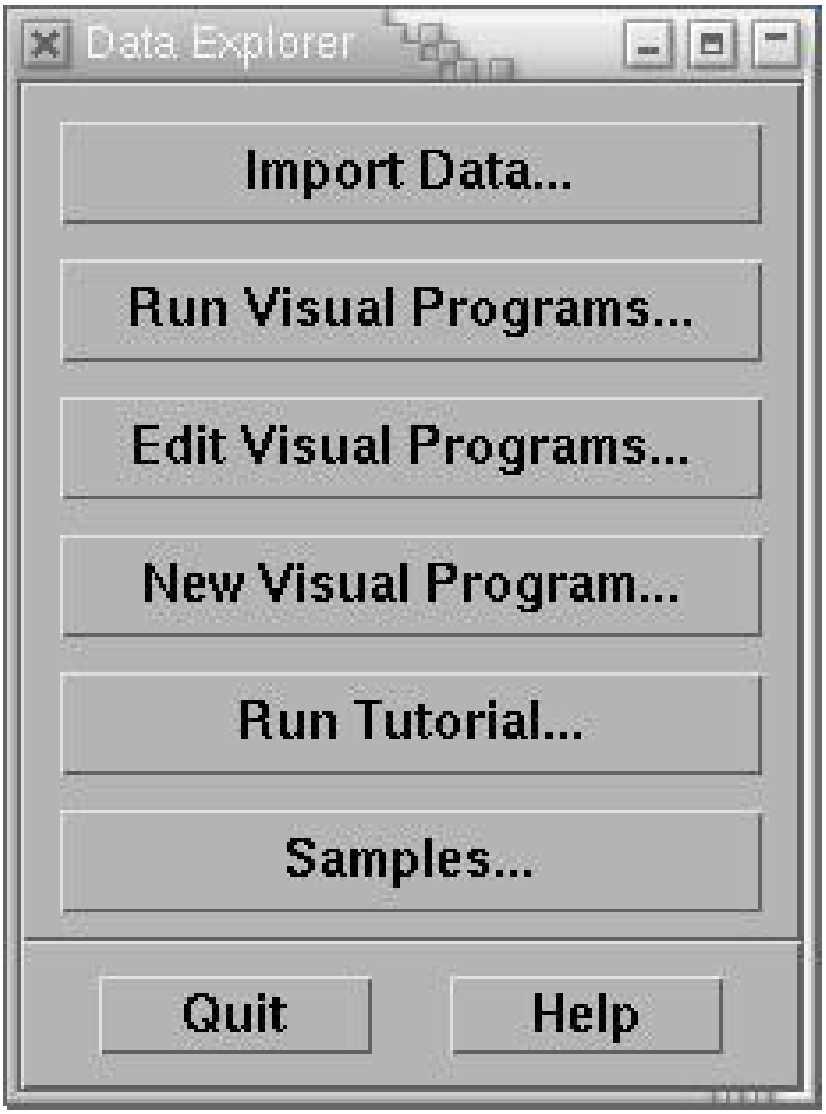


Figura 5-1 Menú OpenDX

Para correr OpenDX solamente es necesario digitar *dx* y aparecerá el menú principal (Figura 5-1).

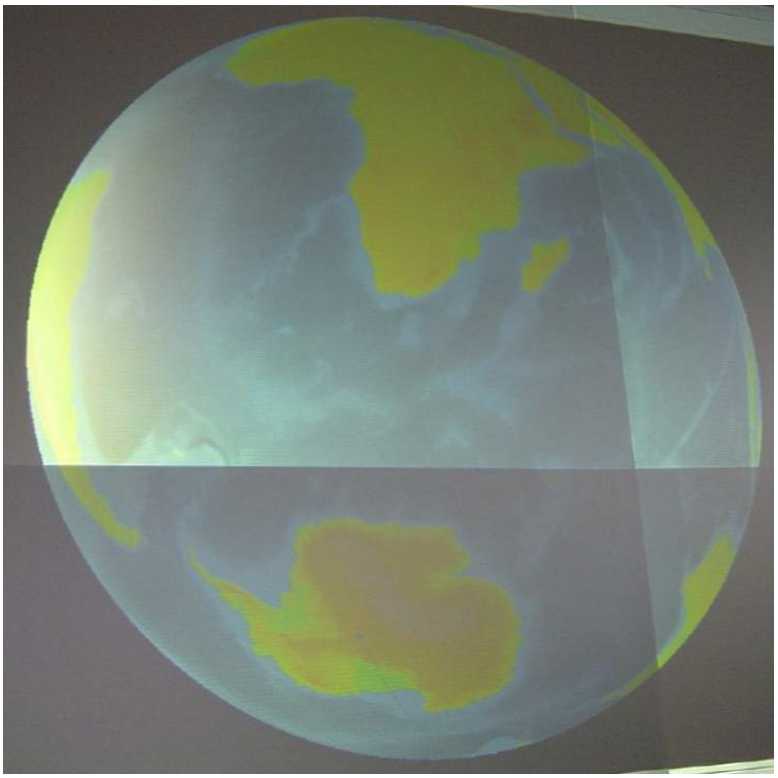


Figura 5-2 Ejemplo globe.net en OpenDX

Para comprobar el funcionamiento de este paquete de software se abrieron y manipularon distintos ejemplos de diversos campos de estudio. El primer ejemplo que se probó fue *globe.net*. Este muestra la topografía de la tierra de acuerdo a datos provistos por la NASA (Figura 5-2).

También se corrió *vortex.net*, que muestra el agujero en la capa de ozono tanto en el polo norte como en el polo sur (Figura 5-3). Este ejemplo provee una animación con los datos de vientos y temperatura entre octubre 17 y noviembre 16 de 1987. Los datos de esta

visualización fueron recogidos por el *National Space Science Data Center* de la NASA.

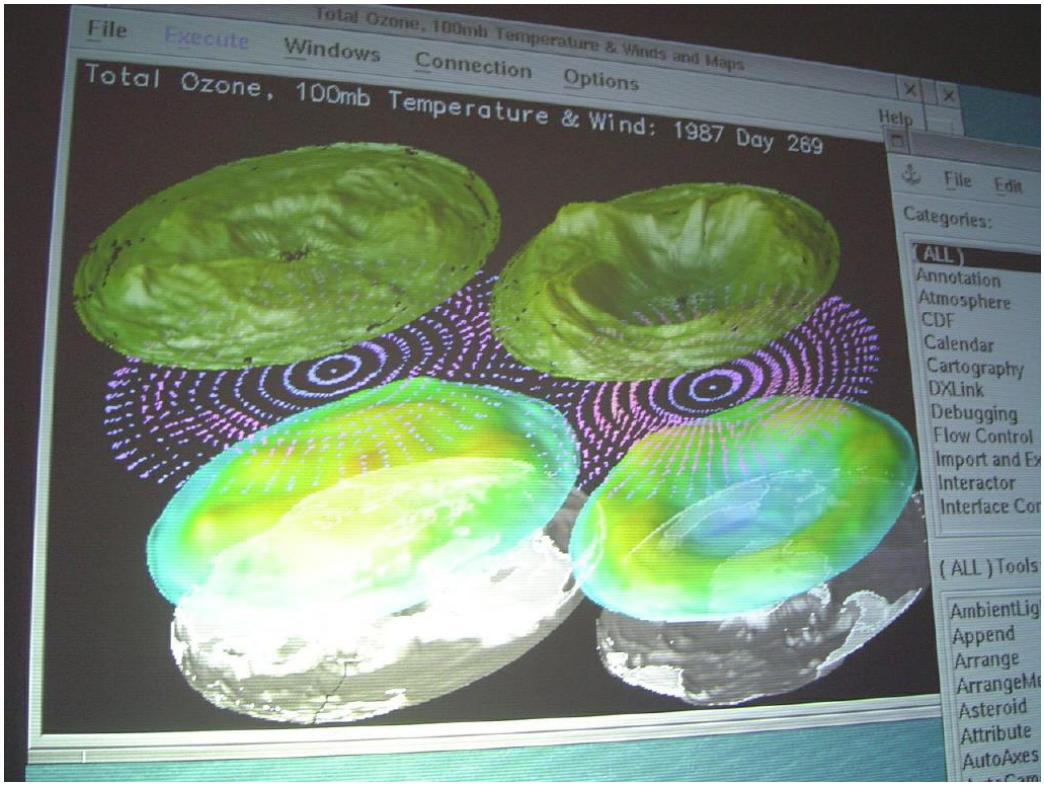


Figura 5-3 Ejemplo vortex.net en OpenDX

Por último se abrió un ejemplo aplicado a la medicina llamado *MRI_2.net*. En este, como muestra la Figura 5-4, se puede ver la resonancia magnética de un cerebro humano como un volumen de tres dimensiones.

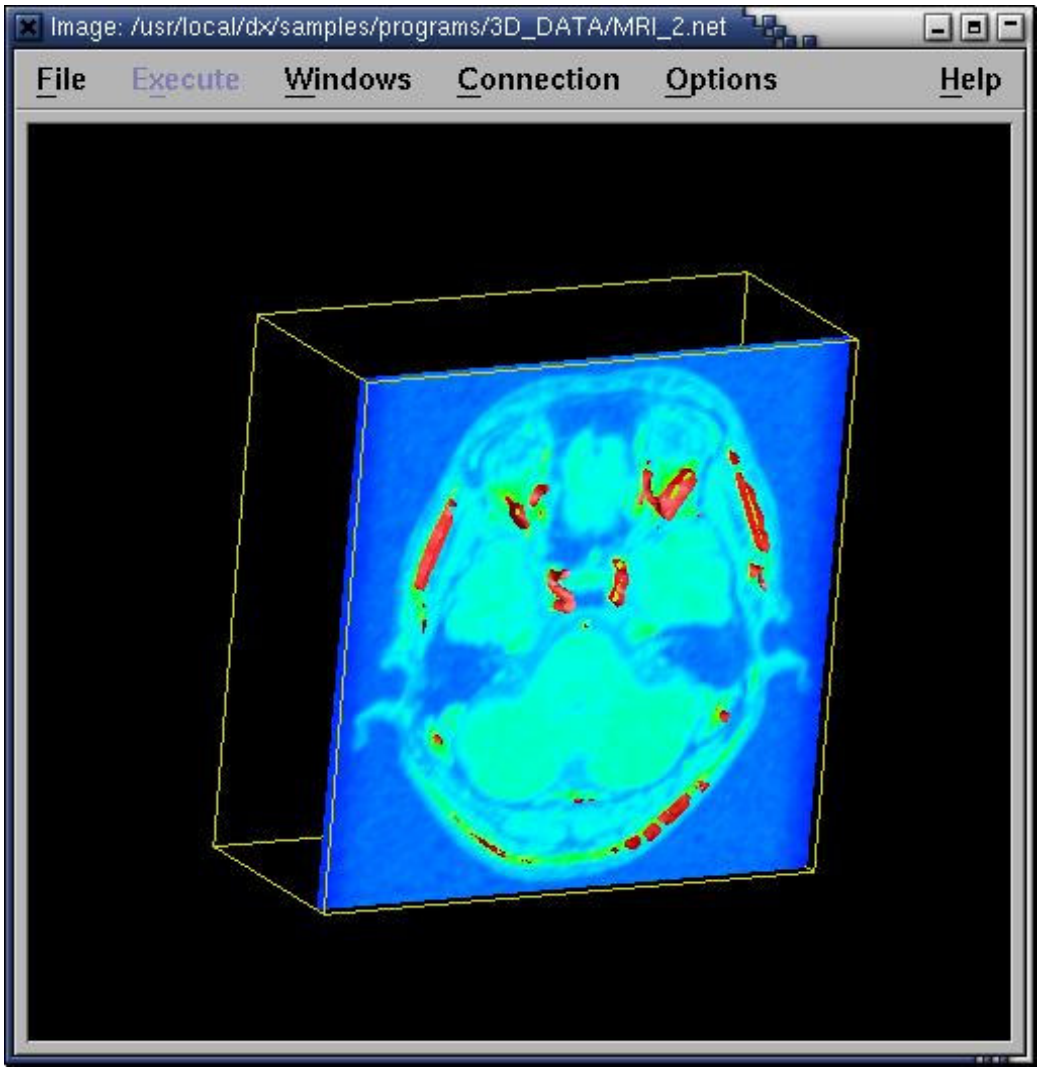


Figura 5-4 Ejemplo MRI_2.net en OpenDX

Este último ejemplo también posee una animación como se puede ver en la Figura 5-5.

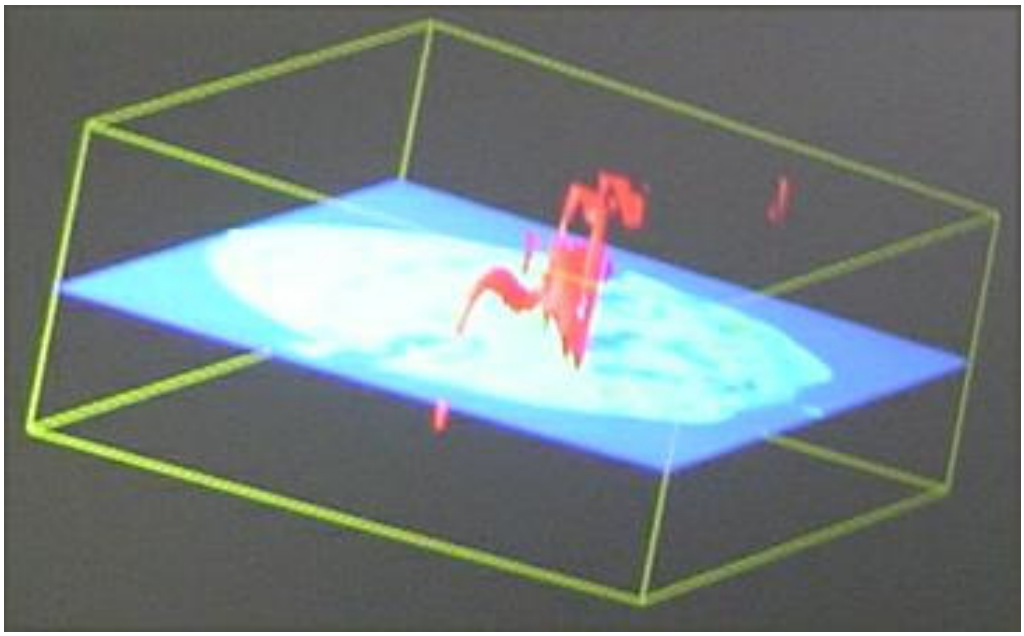


Figura 5-5 Animación del ejemplo MRI_2.net en OpenDX

5.1.2 Vis5d

Otro programa de visualización de datos del clima que se probó es Vis5d+ versión 1.2.1. Para compilarlo e instalarlo se digita:

```
tar -zxvf vis5d+-121tar.gz
./configure
make
src/vis5d hole.v5d -path src
make install
```

Luego para correr el programa se digita *vis5d* y el nombre del archivo que se quiere visualizar. Por ejemplo:

```
vis5d rsmh10_2002092700.v5d
```

Este ejemplo, que se puede ver en la Figura 5-6, es un modelo regional espectral de 10 kilómetros sobre las islas Hawaii.

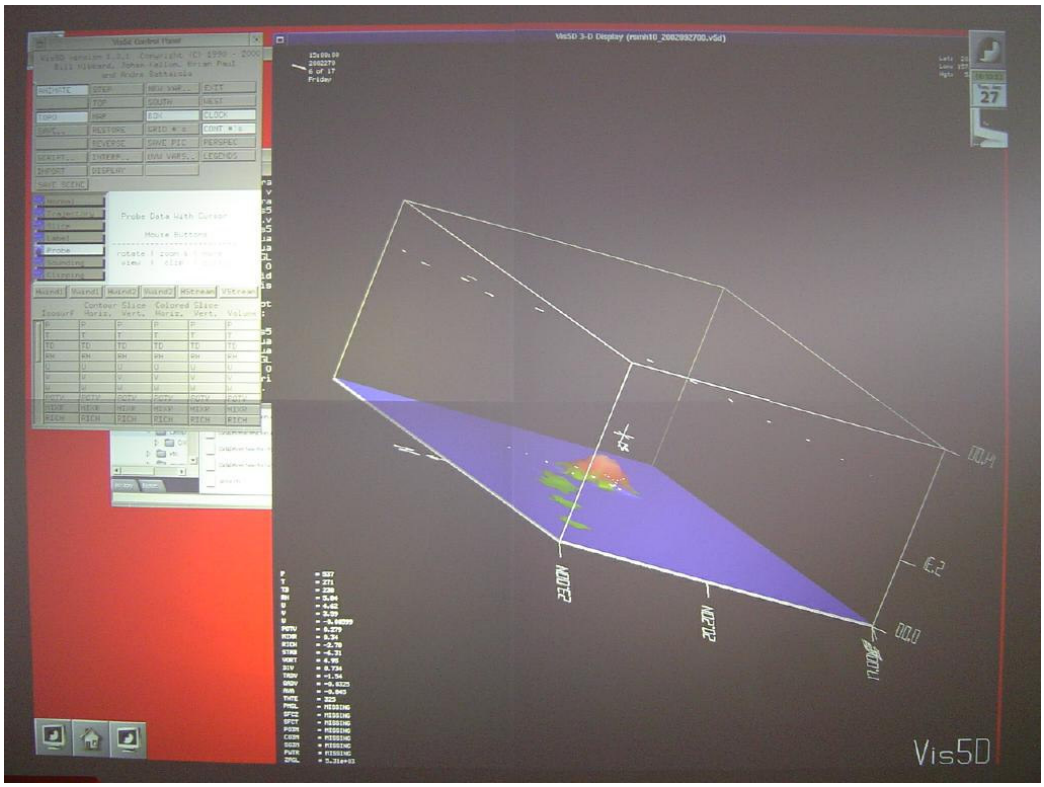


Figura 5-6 Ejemplo de vis5d con una vista sobre las islas Hawaii

En este ejemplo se pueden ver tanto las predicciones de la dirección del viento hasta una altura de diez kilómetros, como de la temperatura en las islas Hawaii y sus alrededores, tal como se aprecia en la Figura 5-7.

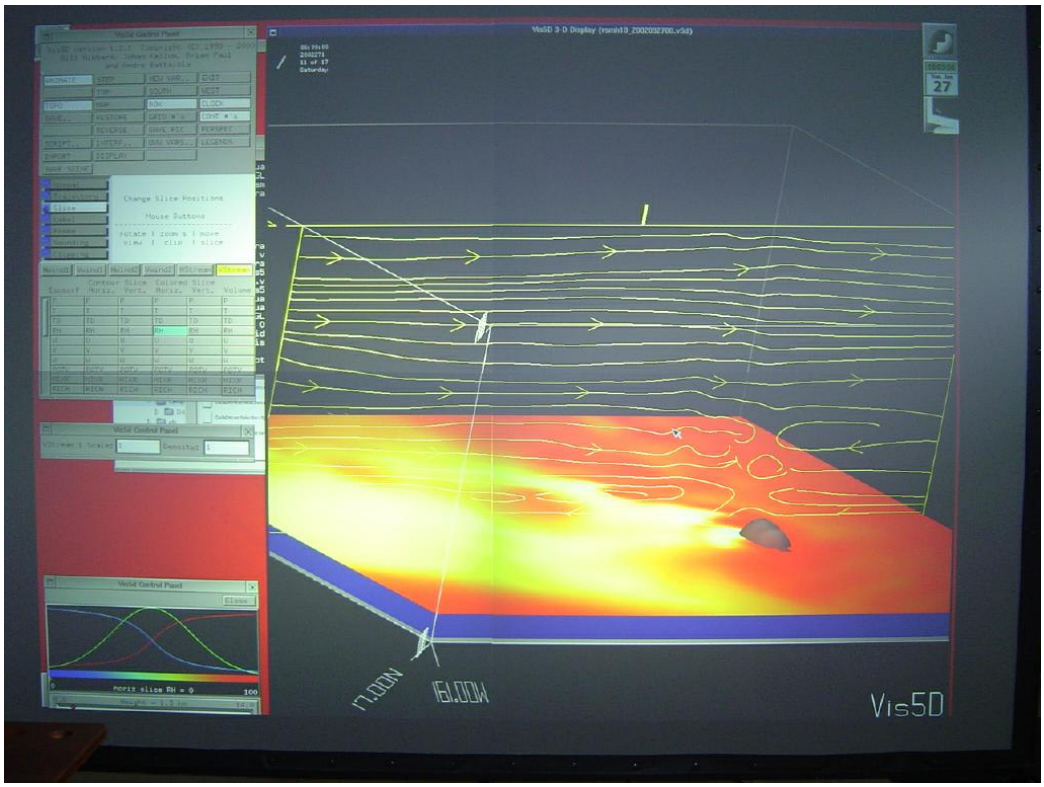


Figura 5-7 Dirección del viento y temperatura en Hawaii

Los datos utilizados en este ejemplos fueron provistos por el *National Centers for Environmental Prediction* de los Estados Unidos de Norteamérica.

5.2 Pruebas de rendimiento

Para probar el rendimiento del sistema se utilizó como parámetro el número de cuadros por segundo que son dibujados. Para obtener los datos se utilizaron tres programas que usan OpenGL: Atlantis, city y glxgears.

La prueba realizada con Atlantis (Figura 5-8), dio como resultado un promedio de 59.7 cuadros por segundo cuando el programa corre en la pantalla completa, es decir a una resolución de 1600 x 1200 píxeles.

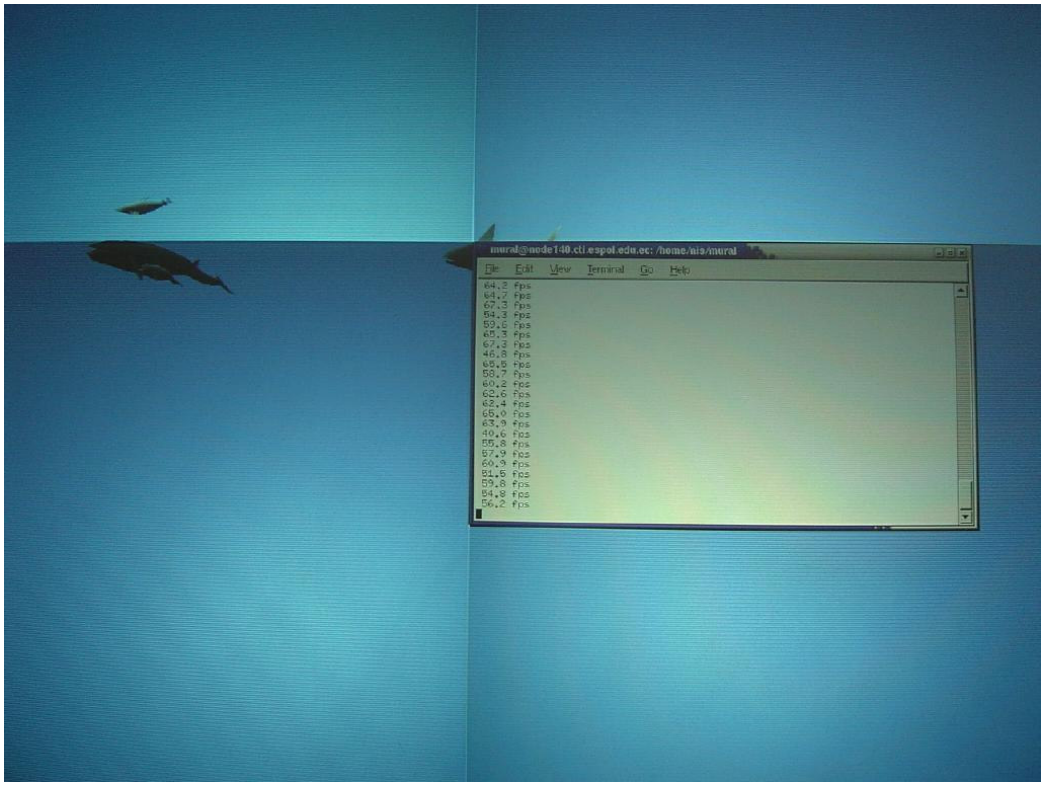


Figura 5-8 Prueba de Atlantis en el sistema distribuido con Chromium

La prueba con city en pantalla completa dio un promedio de 2.7 cuadros por segundo (Figura 5-9).

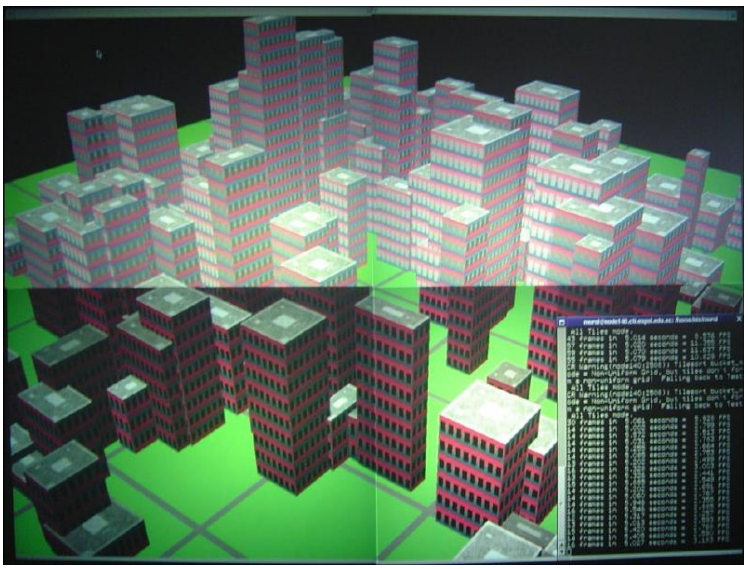


Figura 5-9 Prueba de city en el sistema distribuido con Chromium

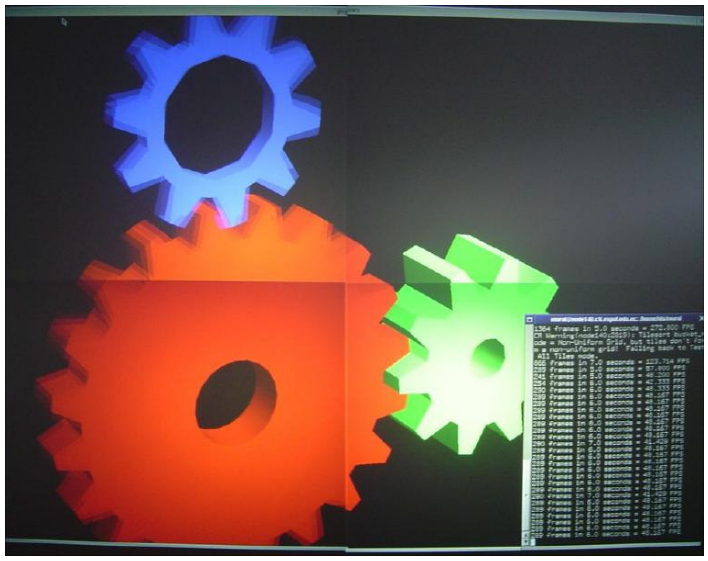


Figura 5-10 Prueba de glxgears en el sistema distribuido con Chromium

Al realizar la prueba con glxgears en pantalla completa (Figura 5-10), dio como promedio 47.1 cuadros por segundo.

Los datos utilizados para sacar estos promedios se encuentran en la sección B.1 del Apéndice B.

5.2.1 Comparación con sistema no distribuido

En esta prueba se comparará el rendimiento del sistema de visualización con el rendimiento de un solo computador conectado a un solo proyector.

La primera gran diferencia que existe entre los dos sistemas es que con un computador solamente podemos alcanzar una resolución nativa de 800 x 600, es decir cuatro veces menor a la del sistema distribuido. Además si se quiere que la proyección alcance el mismo tamaño que tiene el sistema (230 x 172 centímetros), es necesario que la pantalla se encuentre a 320 centímetros del mueble de los proyectores y no a 149 centímetros como cuando se utilizan cuatro proyectores. Esto, además de ocasionar un desperdicio del espacio del cuarto donde se encuentra el sistema de visualización, hace que cada píxel aumente de tamaño a 3.18 x 3.18 milímetros (1/8 de pulgada cuadrada), en lugar de 1.59 x 1.59 milímetros (1/16 de pulgada) que es lo que miden cuando se utilizan cuatro proyectores (Figura 5-11).

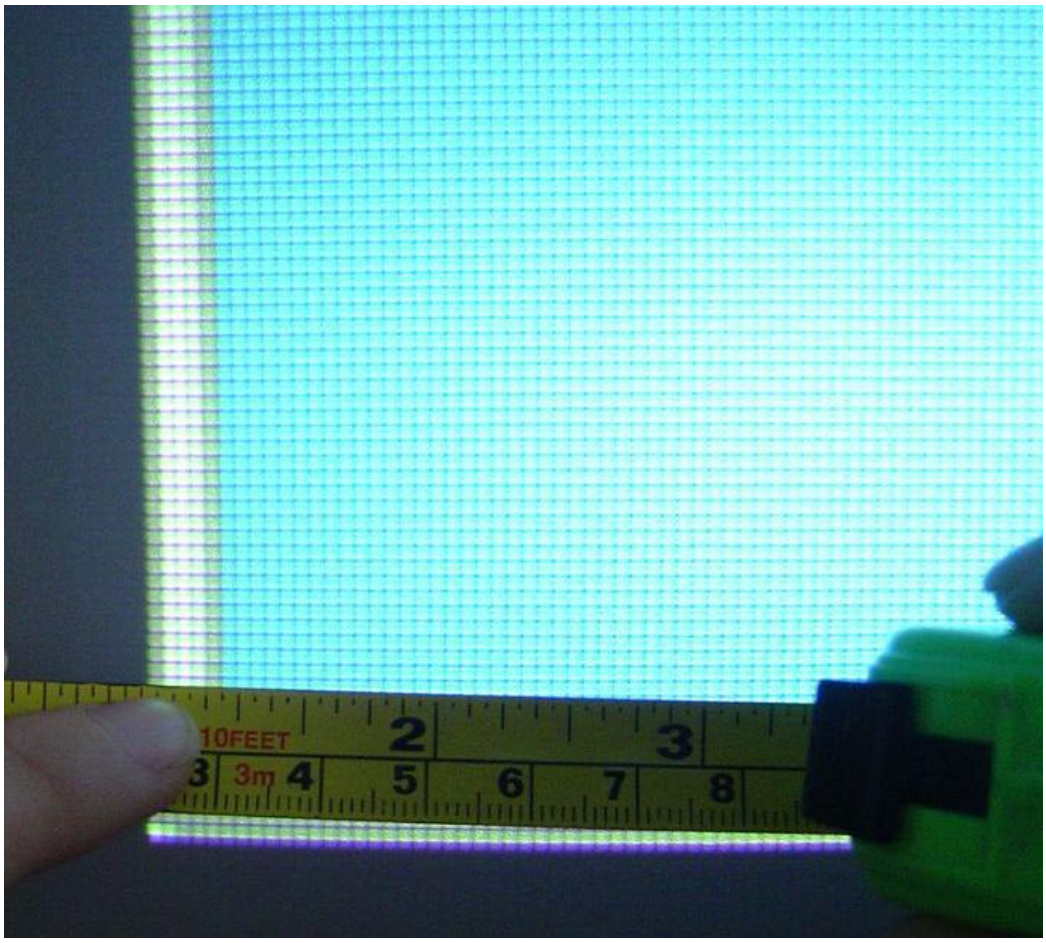


Figura 5-11 Medición de un píxel

En cuanto al rendimiento del sistema, no se puede hacer una comparación precisa ya que como se indicó anteriormente, la resolución máxima que se puede alcanzar utilizando un computador es cuatro veces menor que la del sistema que utiliza cuatro computadores, por lo tanto como tiene que graficar menos píxeles, el rendimiento debería ser mayor en el computador que trabaja solo.

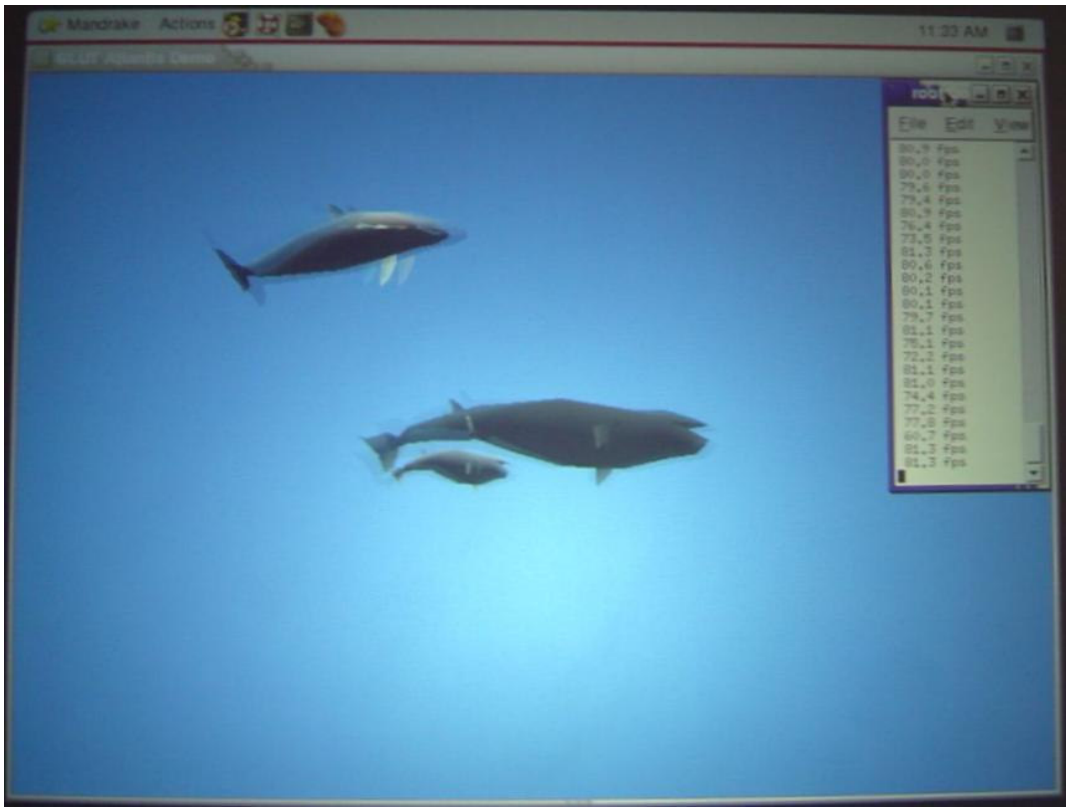


Figura 5-12 Prueba de Atlantis en un computador

Al realizar la prueba con Atlantis (Figura 5-12) se obtiene un promedio de 79.5 cuadros por segundo.

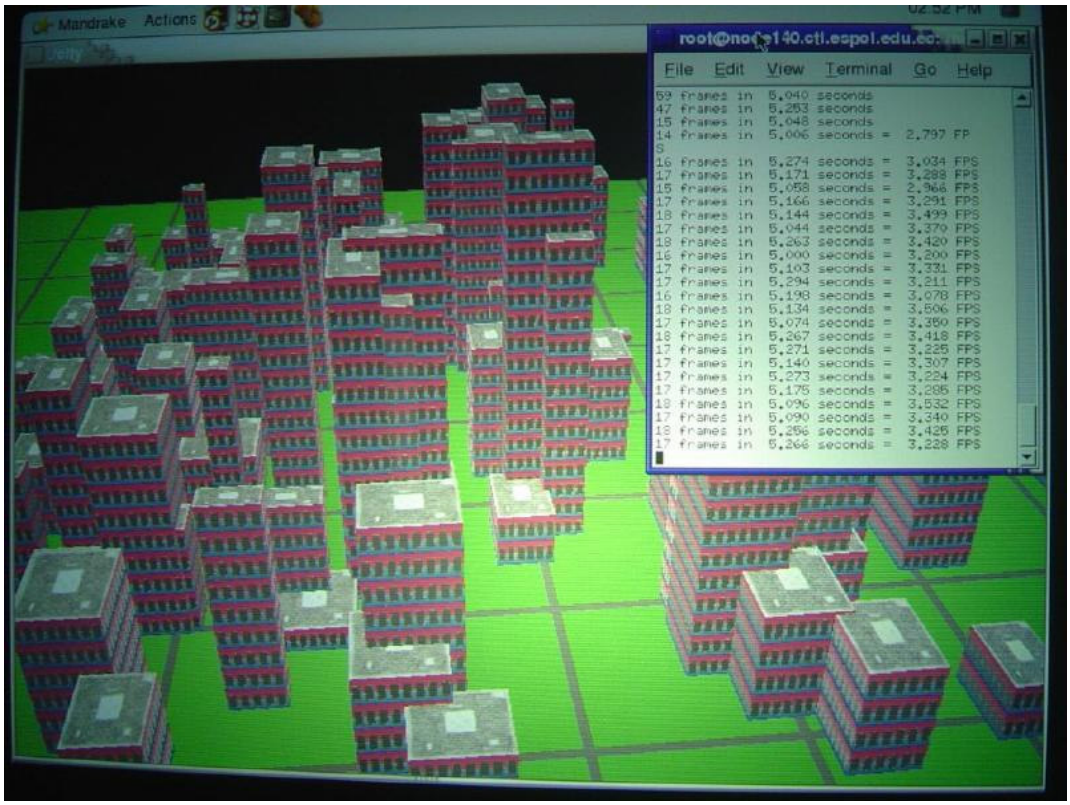


Figura 5-13 Prueba de city en un computador

En cambio al correr city, el promedio es de 3.3 cuadros por segundo (Figura 5-13).

Por último, al realizar la prueba con glxgears (Figura 5-14) el promedio es de 59.2 cuadros por segundo.

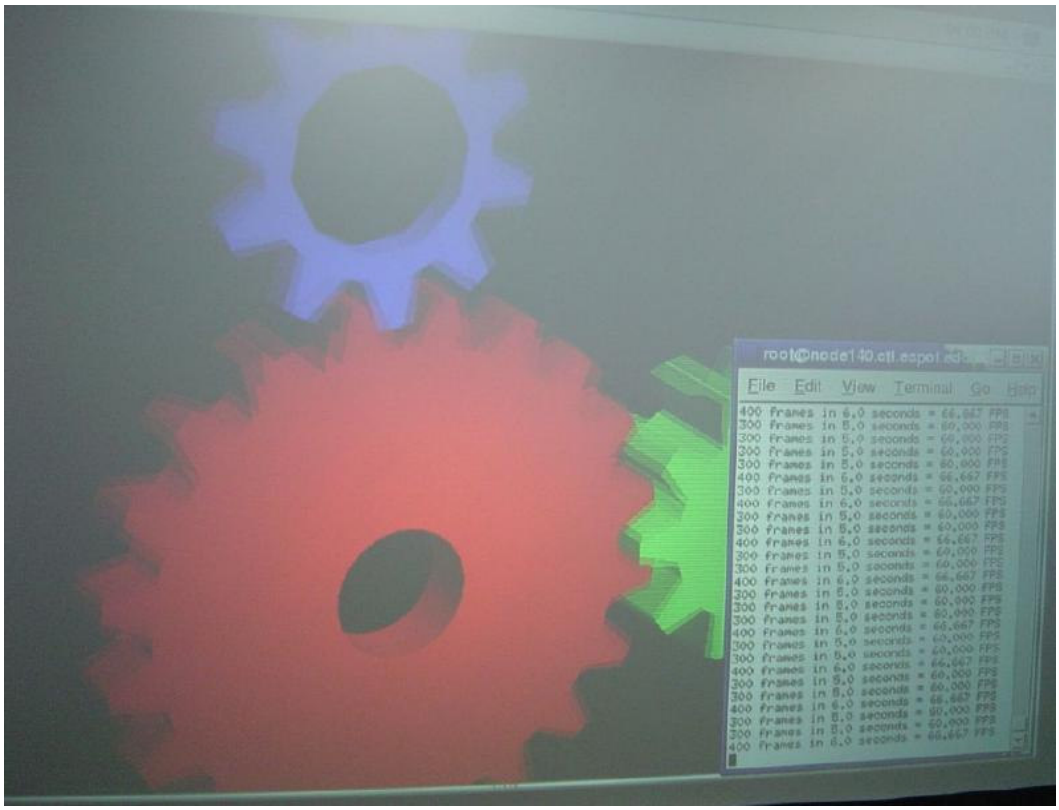


Figura 5-14 Prueba de glxgears en un computador

En la sección B.2 del Apéndice B se pueden observar los datos utilizados para obtener los promedios.

Al comparar los promedios de rendimiento (Tabla 5-1) encontramos que un computador solo es un poco más rápido que el sistema cuando trabaja en pantalla completa, pero no cuatro veces más rápido como indicaría el número de píxeles dibujados. Esto se debe a que el sistema de visualización realiza gran parte del procesamiento gráfico en paralelo y por lo tanto el rendimiento final del mismo aumenta.

Tabla 5-1 Comparación de características y rendimiento de un computador y un sistema distribuido con Chromium

	Un computador	**Sistema distribuido con Chromium**
Tamaño de la proyección	230 x 172 cm	230 x 172 cm
Distancia entre la pantalla y los proyectores	320 cm	149 cm
Resolución máxima	800 x 600 píxeles	1600 x 1200 píxeles
Tamaño de cada píxel	3.18 x 3.18 mm	1.59 x 1.59 mm
Rendimiento (en cuadros por segundo)		
Atlantis	79.5 fps	59.7 fps
city	3.3 fps	2.7 fps
glxgears	59.2 fps	47.1 fps

5.2.2 Comparación con otros sistemas distribuidos

En esta prueba se comparará el rendimiento del sistema con otro que también distribuye el escritorio entre cuatro computadores a través de DMX, pero que no utiliza Chromium.

Al tener cuatro computadores con el escritorio distribuido, la resolución máxima y el tamaño de cada píxel son iguales a los del sistema original. Así mismo se necesita la misma distancia entre los proyectores y la pantalla.

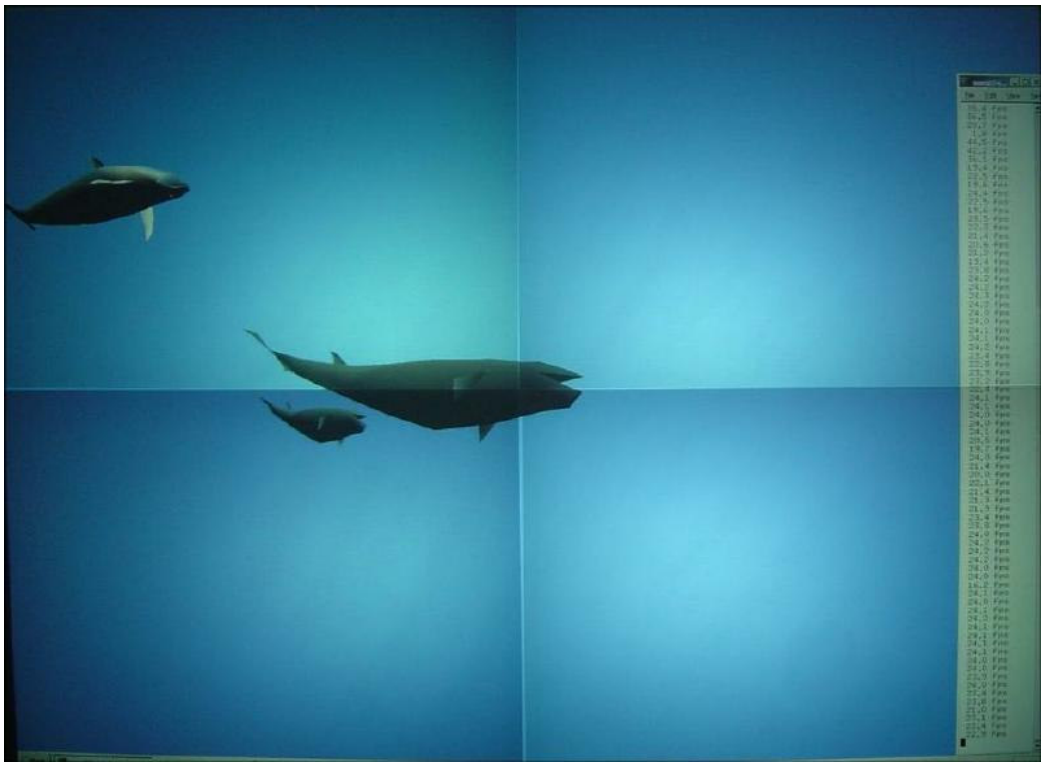


Figura 5-15 Prueba de Atlantis en el sistema distribuido sin Chromium

En lo que sí vamos a notar diferencia es en el rendimiento del sistema. Al realizar la prueba con Atlantis en pantalla completa (Figura 5-15), obtenemos una velocidad promedio de 22.8 cuadros por segundo.

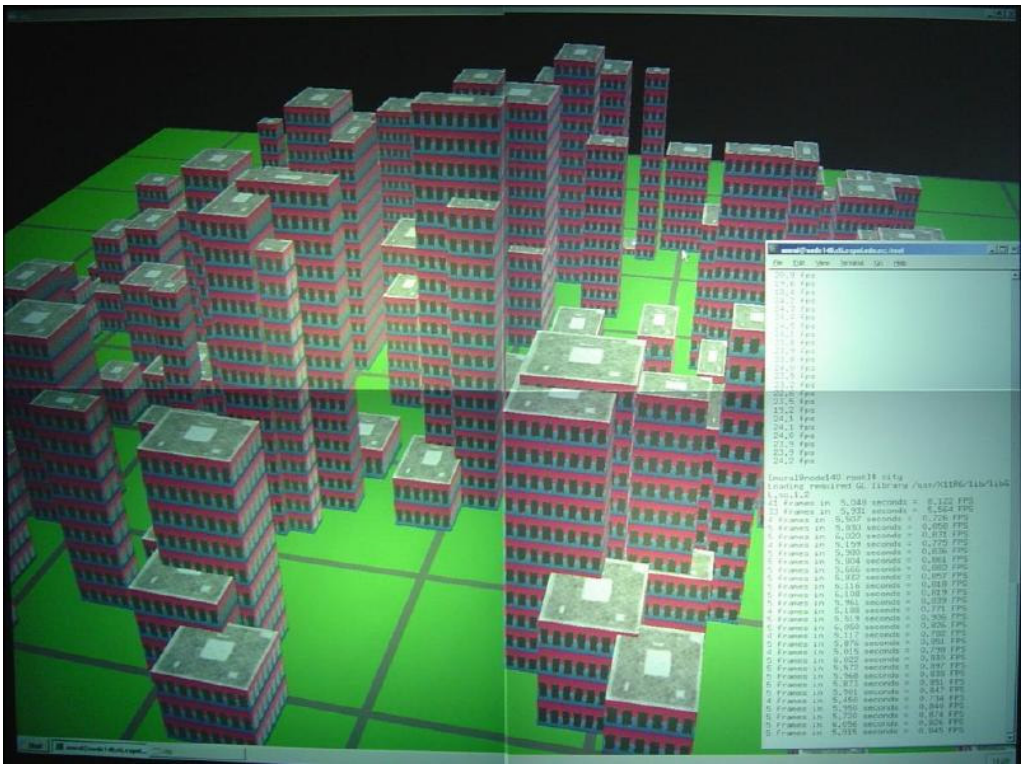


Figura 5-16 Prueba de city en el sistema distribuido sin Chromium

El promedio de cuadros por segundo corriendo city es de 0.8 (Figura 5-16).

Cuando se prueba glxgears en pantalla completa (Figura 5-17), la velocidad promedio es 20.9 cuadros por segundo.

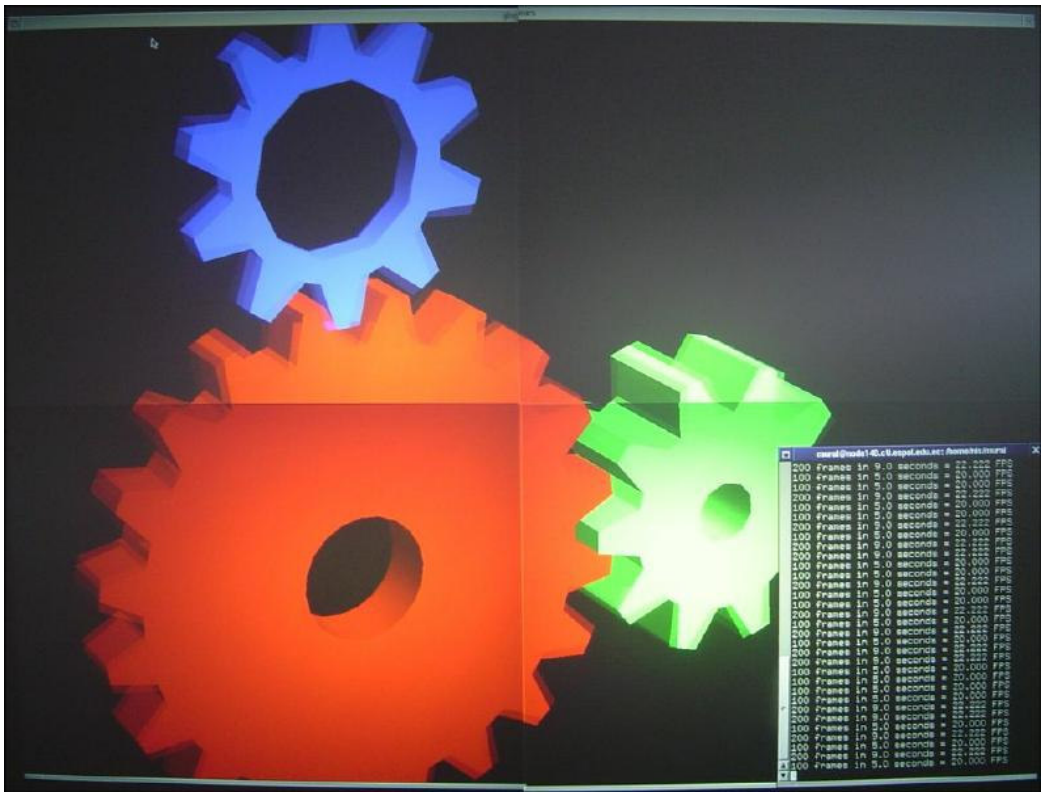


Figura 5-17 Prueba de glxgears en el sistema distribuido sin Chromium

Los datos utilizados para sacar los promedios se encuentran en la sección B.3 del Apéndice B.

Al comparar estos promedios con los obtenidos por el sistema que utiliza Chromium (Tabla 5-2) se observa que la velocidad del sistema que utiliza Chromium es casi el triple que la otra, por lo tanto podemos concluir que se obtiene un mejor rendimiento al distribuir las operaciones OpenGL con Chromium, y no solamente distribuyendo el escritorio con DMX.

Tabla 5-2 Comparación de características y rendimiento de un sistema distribuido sin Chromium y un sistema distribuido con Chromium

	Sistema distribuido sin Chromium	Sistema distribuido con Chromium
Tamaño de la proyección	230 x 172 cm	230 x 172 cm
Distancia entre la pantalla y los proyectores	149 cm	149 cm
Resolución máxima	1600 x 1200 píxeles	1600 x 1200 píxeles
Tamaño de cada píxel	1.59 x 1.59 mm	1.59 x 1.59 mm
Rendimiento (en cuadros por segundo)		
Atlantis	22.8	59.7 fps
city	0.8	2.7 fps
glxgears	20.9	47.1 fps

Si estos datos se comparan con los resultados de un solo computador (Tabla 5-3), se puede notar que el rendimiento es prácticamente cuatro veces menor, lo cual es lógico, ya que el sistema tiene que graficar el cuádruple de píxeles en la pantalla completa.

Tabla 5-3 Comparación de rendimiento de un computador, un sistema distribuido con Chromium y un sistema distribuido sin Chromium (en cuadros por segundo)

	Un computador	Sistema distribuido con Chromium	Sistema distribuido sin Chromium
Atlantis	79.5	59.7	22.8
city	3.3	2.7	0.8
glxgears	59.2	47.1	20.9

5.3 Pruebas de sistema para colaboración y control remoto de aplicaciones

Para la colaboración y control remoto de aplicaciones se utilizó el VNC Viewer versión 3.3.7. Para instalarlo, en una ventana de comandos se digita:

```
tar -zxvf vnc-3.3.7-x86_linux.tar.gz
```

Para correrlo solamente se digita en una consola *vncviewer*.

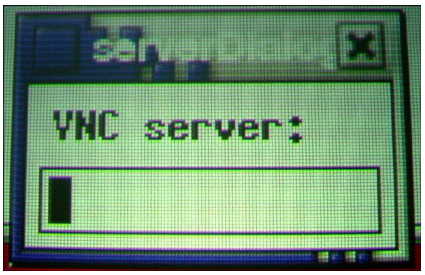


Figura 5-18 Ventana para poner la dirección IP del servidor VNC

Aparece una ventana que pide la dirección IP del computador que se quiere controlar (Figura 5-18).

El computador que va a ser controlado debe tener instalado y estar corriendo el servidor VNC. Luego de ingresar la contraseña, si el servidor VNC así lo requiere, aparecerá una ventana en la cual se podrá manipular el escritorio del computador remoto.

Como se puede ver en la Figura 5-19 y la Figura 5-20, en las pruebas que se realizaron, el sistema controló remotamente el escritorio de un computador que tenía instalado Windows XP.

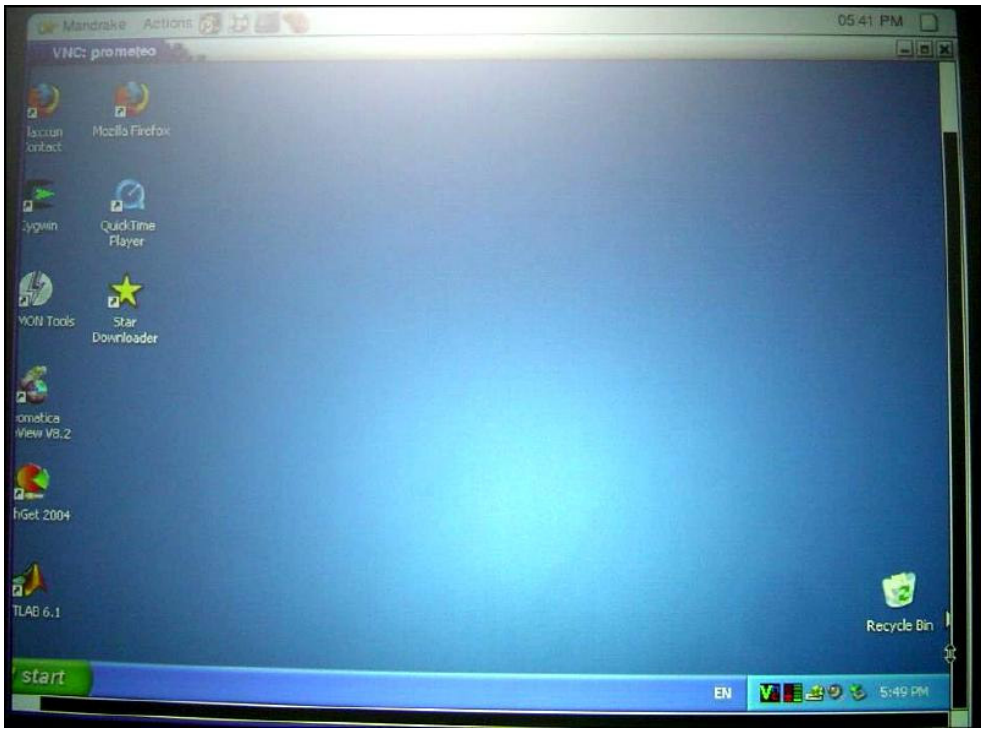


Figura 5-19 Ventana para poner la dirección IP del servidor VNC

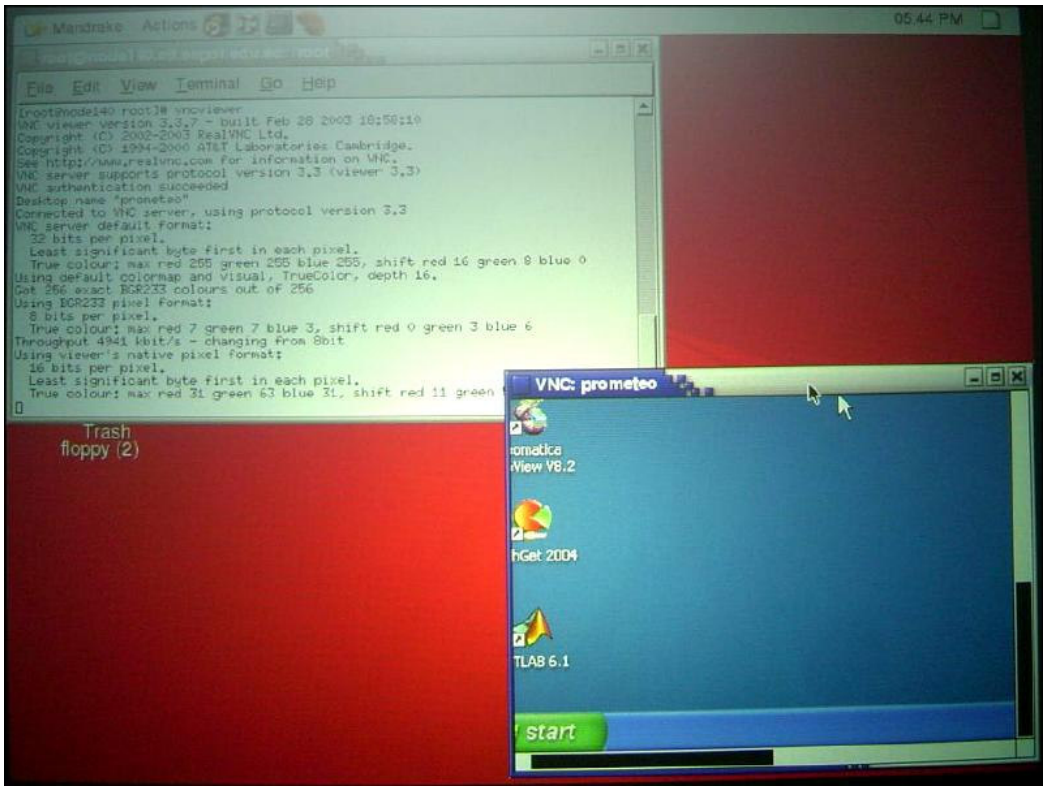


Figura 5-20 Ventana para poner la dirección IP del servidor VNC

Cabe indicar que cuando se controla un computador remotamente, el sistema no utiliza Chromium y por lo tanto no se distribuyen las instrucciones gráficas, lo cual como se demostró en la sección 5.2 conlleva un rendimiento menor.

5.4 Uso del sistema para visualización de datos reales obtenidos en la ESPOL

Para probar el sistema, se realizó una visualización de la pluviosidad en milímetros de la costa ecuatoriana entre enero de 1982 y diciembre de 1983, en la cual se puede apreciar como durante los meses de invierno el Fenómeno del Niño provocó un aumento considerable de la cantidad de agua que cayó sobre este sector.

Los datos con los cuales se realizó esta prueba fueron provistos por el doctor José Luis Santos, quien es un profesor e investigador de la ESPOL y además es el director del Centro Internacional de Investigaciones sobre el Fenómeno de El Niño (CIIFEN). Estos datos se encuentran en el Apéndice C, y comprenden información de pluviosidad en cinco ciudades de la costa ecuatoriana: Esmeraldas, Bahía de Caráquez, Manta, Portoviejo y Guayaquil.

Para esta prueba se utilizó OpenDX y se realizaron dos programas, el primero mostraba los resultados en dos dimensiones, y el otro los mostraba en tres dimensiones, utilizando la altura de las ciudades donde fueron recogidos los datos.

La estructura del primer programa se puede ver en la Figura 5-21, y uno de sus resultados en la Figura 5-22.

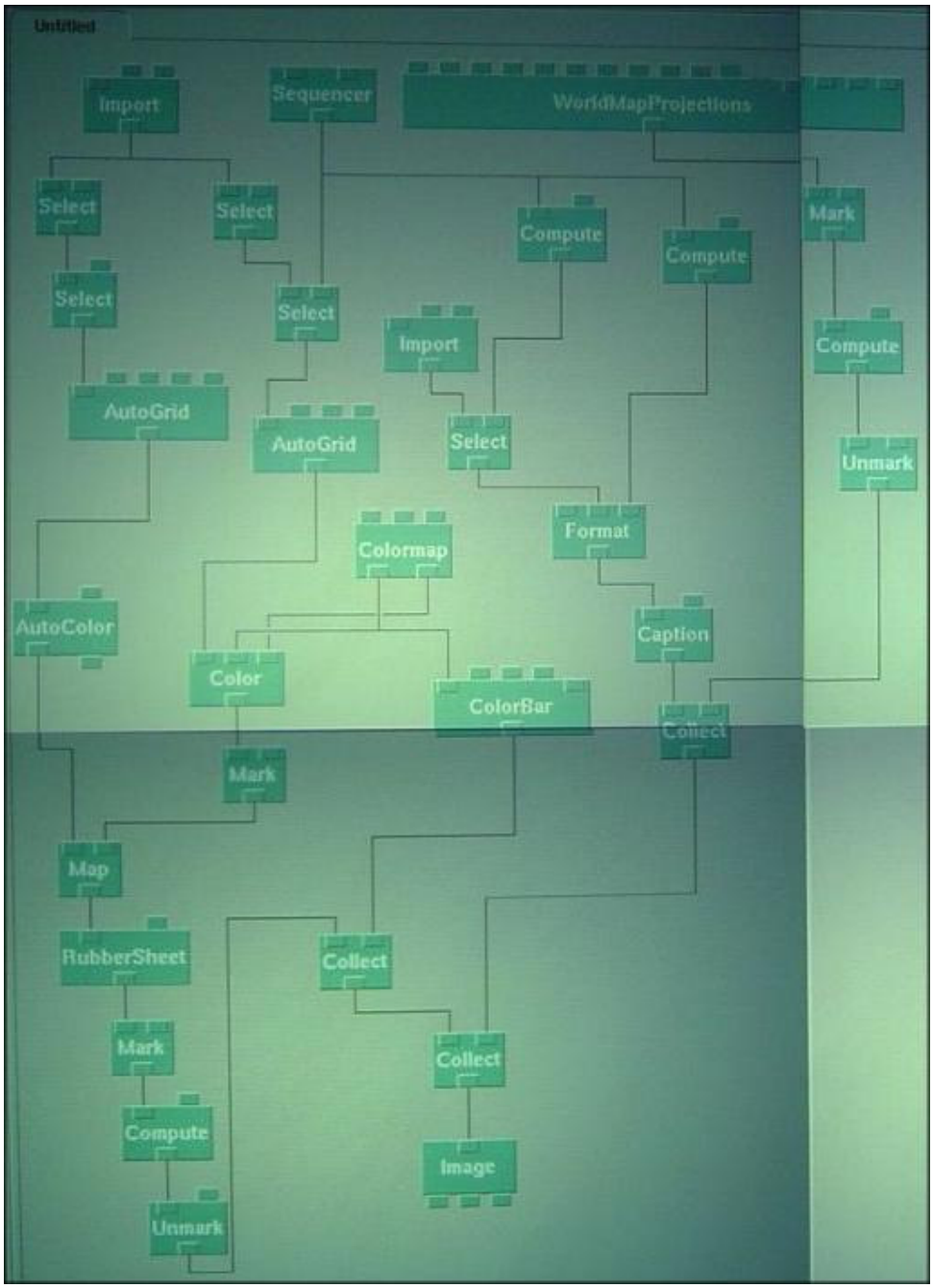


Figura 5-21 Programa que muestra los resultados de pluviosidad en dos dimensiones

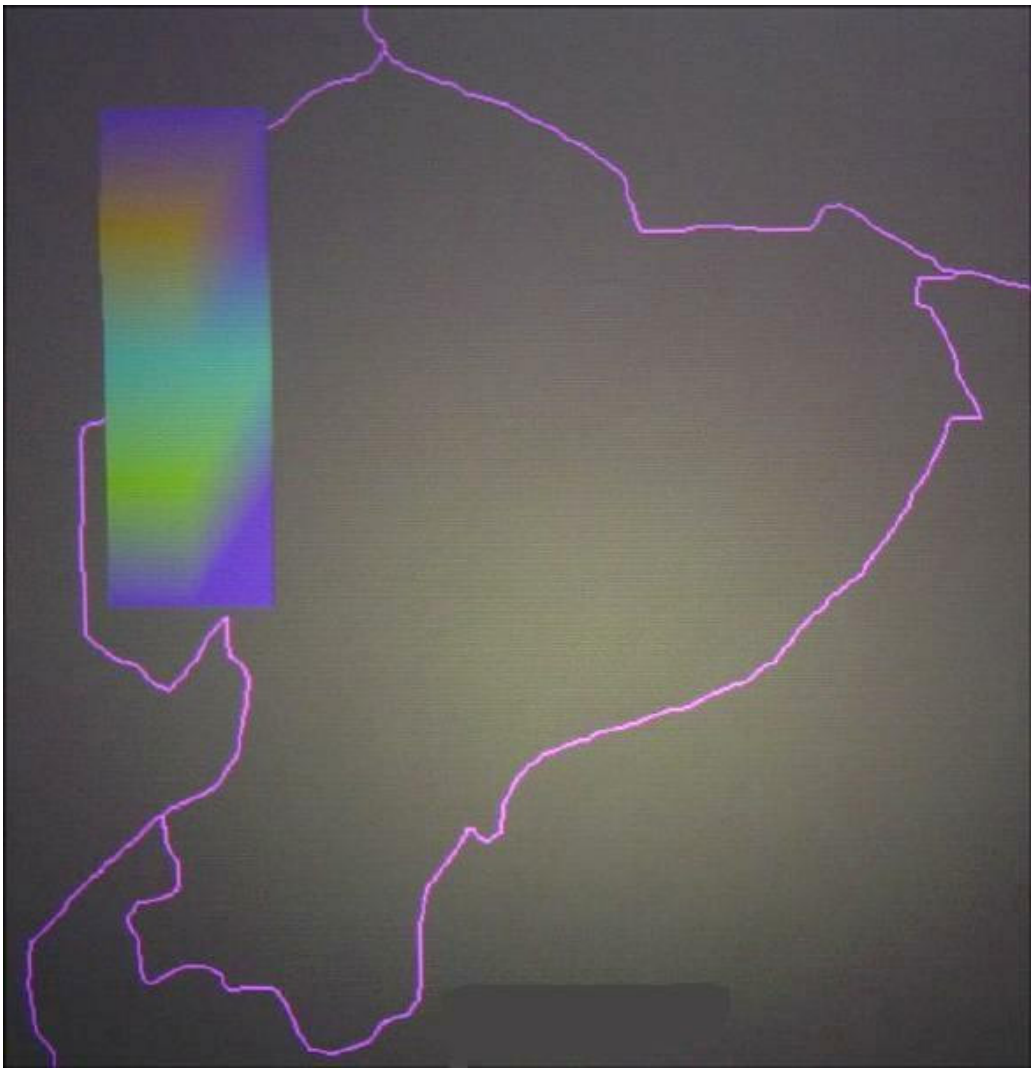


Figura 5-22 Pluviosidad de la costa ecuatoriana en dos dimensiones

El programa que muestra la pluviosidad en la costa ecuatoriana en tres dimensiones quedó como se muestra en la Figura 5-23 y uno de sus resultados se aprecia en la Figura 5-24.

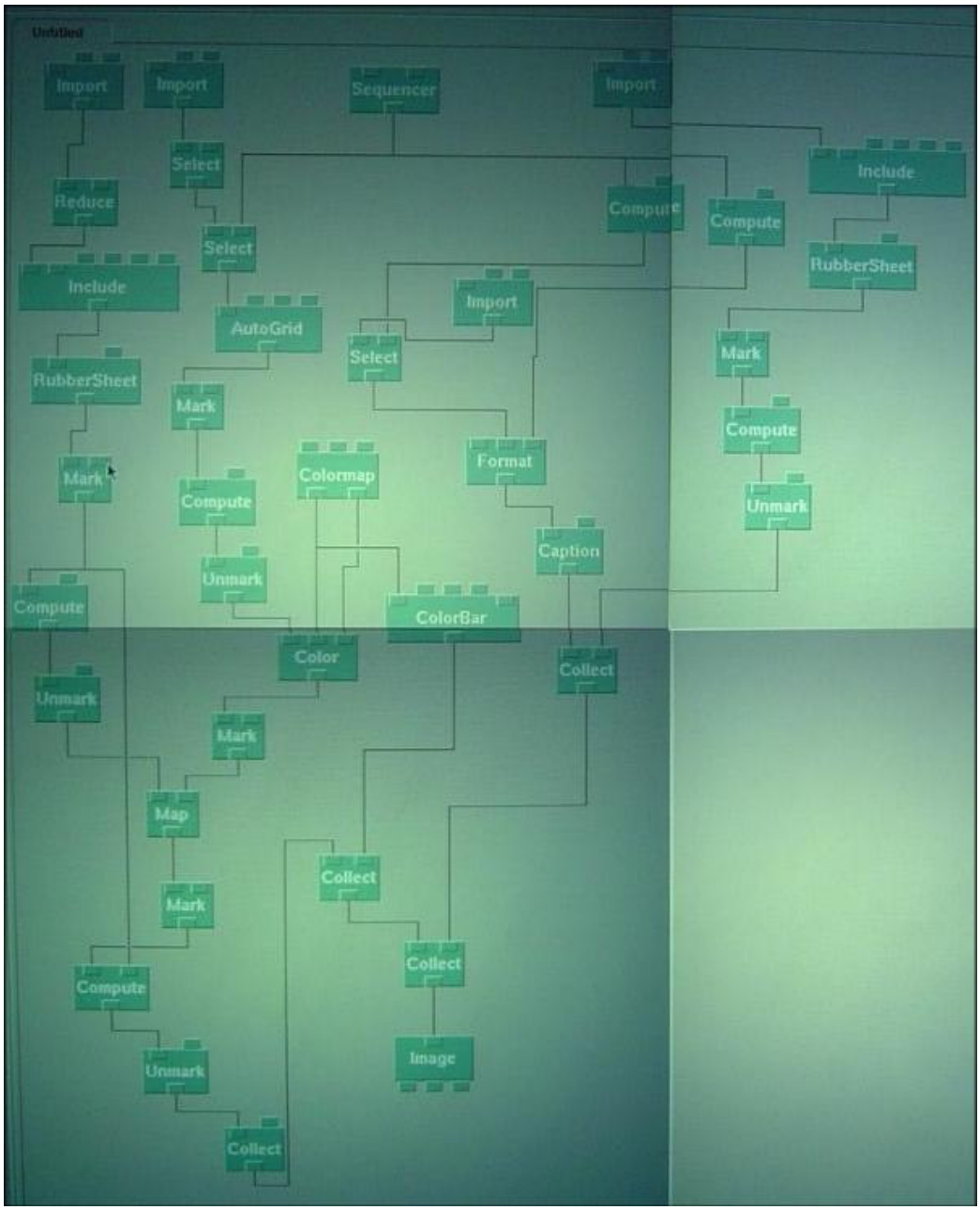


Figura 5-23 Programa que muestra los resultados de pluviosidad en tres dimensiones

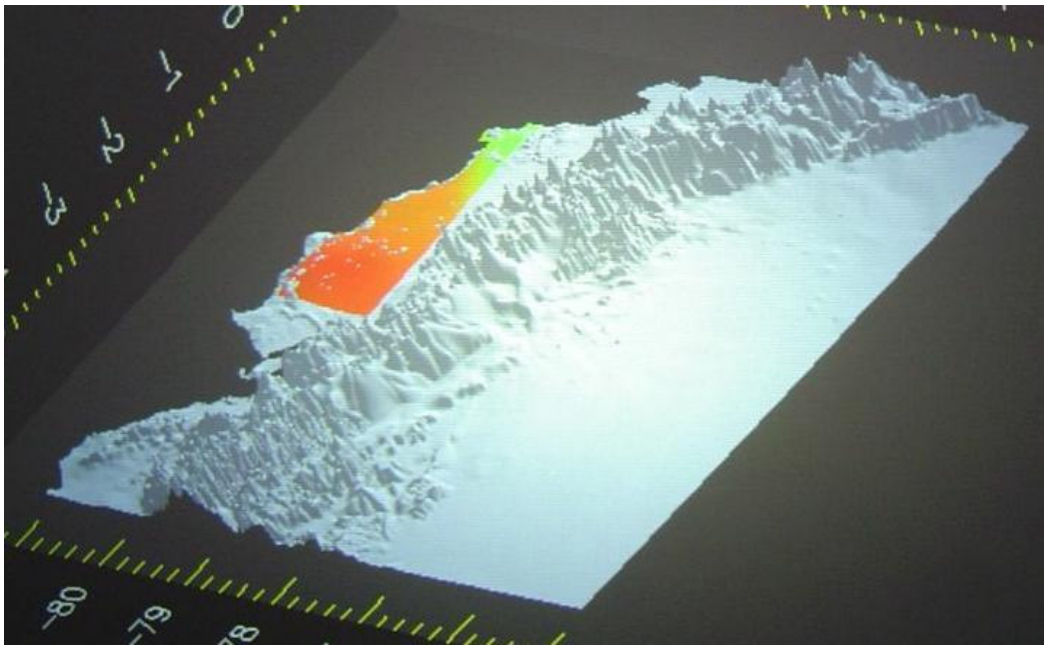


Figura 5-24 Pluviosidad de la costa ecuatoriana entre enero de 1982 y diciembre de 1983 en tres dimensiones

Cabe recalcar que los dos programas cuentan con animaciones en las cuales van cambiando los colores que muestran el nivel de pluviosidad a medida que avanzan los meses. Los resultados de la animación en tres dimensiones se pueden ver en el Apéndice D.

CONCLUSIONES Y RECOMENDACIONES

5.5 Conclusiones

- Es factible construir en la ESPOL un sistema de visualización científica de alta resolución a un costo relativamente bajo y con resultados aceptables.
- De acuerdo a la forma en que fue diseñada y construida la pantalla, y a la capacidad de los programas que se utilizaron para el sistema de visualización, este es fácilmente escalable a más nodos y proyectores, con lo cual se obtendría una resolución mucho mayor a la del sistema actual.
- La utilización de proyectores de 1,500 lúmenes y de una pantalla de proyección trasera permiten que el sistema pueda ser utilizado en un cuarto con las luces encendidas y que el usuario pueda caminar frente a la pantalla sin interferir con la proyección.
- El software de distribución libre disponible en la actualidad permite configurar un sistema de visualización. Sin embargo, los programas utilizados, al estar todavía en desarrollo, presentan pequeños problemas al momento de configurar el sistema. La solución a estos problemas fue descrita en esta tesis.

- La distribución de las operaciones gráficas de tres dimensiones (a través de Chromium) permiten que el sistema tenga un rendimiento mucho mayor a que si solamente se distribuyen las operaciones de dos dimensiones.
- El sistema puede acceder y controlar, a través de la red, otros computadores, los cuales pueden estar corriendo un sistema operativo diferente, lo cual permite ver en la pantalla visualizaciones que se obtienen con software que no se puede instalar en Linux, aunque sin la mejora del rendimiento que ofrece Chromium.
- La alineación de los proyectores es un trabajo que necesita mucho tiempo y precisión, por lo cual es necesario que los soportes permitan ajustes finos y tengan por lo menos cuatro grados de libertad. Además, la pantalla debe estar en un lugar fijo.

5.6 Recomendaciones

- Difundir entre los profesores e investigadores de la ESPOL el servicio del sistema de visualización en la universidad, para que el mismo sea aprovechado a cabalidad. Esta difusión se puede realizar a través de cursos o talleres.
- Facilitar la interacción con el usuario utilizando un escritorio y programas que se comporten de una forma parecida a Windows y sus utilitarios, ya que este es el sistema operativo que saben utilizar la mayoría de los usuarios potenciales.
- Buscar software que se pueda aplicar en diversos campos tales como química, física, estadística, entre otros, para que puedan ser utilizados por múltiples áreas de investigación de la ESPOL.
- Aumentar el número de nodos y proyectores a ocho o dieciséis, con lo cual se obtendría una resolución de 3200 x 2400 píxeles o 6400 x 4800, respectivamente.
- Obtener tarjetas de vídeo de mayor velocidad para mejorar el rendimiento del sistema.
- Desarrollar herramientas de colaboración remota más especializadas para que los investigadores locales puedan intercambiar ideas y opiniones con investigadores que se encuentren en cualquier otro punto geográfico.

- Mejorar el sistema de soporte para poder lograr una alineación más precisa de los proyectores, o tratar de encontrar soluciones que corrijan la alineación a través de software, tales como el DeskAlign, descrito por Grant Wallace y sus colegas en el artículo “DeskAlign: Automatically Aligning a Tiled Windows Desktop:” [61]
- Trasladar el sistema a una habitación más grande para que pueda ser utilizado tanto en clases como en charlas y exposiciones.

APÉNDICES

A APÉNDICE A: ARCHIVOS DE CONFIGURACIÓN

A.1 clusterserver.conf

```
#/etc/

NORMAL=`echo -en "\\033[0;32m"`
SUCCESS=`echo -en "\\033[1;32m"`
INFO=`echo -en "\\033[1;34m"`
WARNING=`echo -en "\\033[1;33m"`
WHITE=`echo -en "\\033[0;39m"`

CLIC_INTERFACE=eth0
EXTERNAL_INTERFACE=eth1

###################################
# setup_dns_server VAR
###################################
INTERFACEDNS="$CLIC_INTERFACE"
IPSERVER=`/sbin/ifconfig ${INTERFACEDNS} | grep "inet ad" | sed -e
"s/\\ Bcast.*$//; s/.*://" | sed -e "s/ //"`
IPSERVERREVERSE="`echo $IPSERVER | awk -F. '{print $4}'`"
NAMEOFSERVER=`hostname | awk -F. '{print $1}'`
IPOFFORWARDER=""
TEXTINFO="CLIC_dns_server"
REVERSEIP="`echo $IPSERVER | awk -F. '{print $3"."$2"."$1}'`"
NORMIP="`echo $IPSERVER | awk -F. '{print $1"."$2"."$3}'`"
STARTNODE="141"
FINISHNODE="143"
NAMED_DIR="/var/named"
ZONE_DIR="${NAMED_DIR}/zone"
64 -n HOST mykey`.key"
DNSKEY=""
SERIAL="`date +%d%m%y`01"
ADDSEARCH=""

#####################################
# common VAR for all scripts
#####################################
DATE=`date +%d-%m-20%y`
NETWORKFILE="/etc/sysconfig/network"
NODENAME="node"
DOMAINNAME=`dnsdomainname`
NISDOMAIN=${DOMAINNAME}
INITRD="/etc/rc.d/init.d"
SHORTHOSTNAME=`hostname | awk -F. '{print $1}'`
SBIN_PATH=/usr/sbin
```

```
TFTPSERVER="$IPSERVER"
INTERFACE_DHCP="$CLIC_INTERFACE"
#IPGW=`route -n | grep default | grep eth0 | awk '{print $2}'`
IPGW="$IPSERVER"
NTPSERVER="$IPSERVER"
NODESFILE="/etc/nodes_list"
REP_SAVE="/home/backup"
RESCUE="rescue_stage2"
RESCUE2="${RESCUE}.bz2"

###################################
# setup_server_clic VAR
###################################
USER_SSH_DIR="/root/.ssh"
KEY_SSH_PUB="id_dsa.pub"
KEY_SSH="`echo ${KEY_SSH_PUB} | awk -F. '{print $1}'`"
KEY_AUTH="auth_pub.pem"
LAM_DIR="/etc/lam"
LAM_NODE="lam-bhost.def"
LAM_NODES_FILE="${LAM_DIR}/${LAM_NODE}"
MPI_COMPUTER="machines.LINUX"
MPICH_DIR="/usr/share/mpich/"
TFTPDIR="/var/lib/tftpboot"
NTPFILE="ntpserver"
GMETAD_SOURCES="/etc/gmetad.conf"

###################################
# setup_nis_server var
###################################
NISSERVER="${HOSTNAME}"
HOMENIS="/home/nis"
NIS_MAKEFILE="/var/yp/Makefile"
NFSSERVER="${HOSTNAME}"

###################################
# setup_install_server server
###################################
CDROM="/mnt/cdrom"
INSTALLDIR="/var/install/clic"
MEDIA="cdrom"   # cdrom ftp nfs

###################################
# setup_pbs_server
###################################
PBS_HOME="/var/spool/pbs"
SERVERNAME=${HOSTNAME}
VERSION="2.3.16"
USERADMIN="root"
NODEADMIN=${HOSTNAME}

###################################
# postfix
###################################
MYHOSTNAME="$HOSTNAME"
```

```
MYDOMAIN="$DOMAINNAME"
```

A.2 XF86Config

```
#/etc/X11/

Section "Files"
    FontPath "unix/:-1"
EndSection

Section "ServerFlags"
    AllowMouseOpenFail
EndSection

Section "Keyboard"
    Protocol "Standard"
    XkbModel "pc105"
    XkbLayout "es"
    XkbOptions ""
EndSection

Section "Pointer"
    Protocol "PS/2"
    Device "/dev/psaux"
    Emulate3Buttons
    Emulate3Timeout 50
EndSection

Section "Monitor"
    Identifier "monitor1"
    VendorName "Generic"
    ModelName "800x600 @ 60 Hz"
    HorizSync 31.5-37.9
    VertRefresh 50-70
    # 1024x480 @ 85.6 Hz, 48 kHz hsync
    ModeLine "1024x480" 65.00 1024 1032 1176 1344   480  488  494
563 -hsync -vsync
    # 768x576 @ 79 Hz, 50 kHz hsync
    ModeLine "768x576" 50.00  768  832  846 1000  576  590  595  630
    # 768x576 @ 100 Hz, 61.6 kHz hsync
    ModeLine "768x576" 63.07  768  800  960 1024  576  578  590  616
    # 800x600 @ 56 Hz, 35.15 kHz hsync
    ModeLine "800x600"  36  800  824  896 1024   600  601  603  625
    # 800x600 @ 60 Hz, 37.8 kHz hsync
    ModeLine "800x600"  40  800  840  968 1056   600  601  605  628
+hsync +vsync
    # 800x600 @ 72 Hz, 48.0 kHz hsync
    ModeLine "800x600"  50  800  856  976 1040   600  637  643  666
+hsync +vsync
EndSection

Section "Device"
```

```
    Identifier "device1"
    VendorName "SiS"
    BoardName "SiS 300"
    Option "power_saver"
EndSection

Section "Screen"
    Driver "svga"
    Device "device1"
    Monitor "monitor1"
    DefaultColorDepth 16

    Subsection "Display"
        Depth 8
        Modes "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth 15
        Modes "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth 16
        Modes "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth 24
        Modes "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth 32
        Modes "800x600" "640x480"
    EndSubsection
EndSection

Section "Module"
    Load "GLcore"
    Load "glx"
    Load "extmod"
EndSection
```

A.3 XF86Config-4

```
#/etc/X11/

Section "InputDevice"
    Identifier "Keyboard1"
    Driver "Keyboard"
    Option "XkbModel" "pc105"
    Option "XkbLayout" "la"
    Option "XkbOptions" ""
EndSection
```

```
Section "Monitor"
    Identifier "monitor1"
    VendorName "Generic"
    ModelName "800x600 @ 60 Hz"
    HorizSync 31.5-37.9
    VertRefresh 50-70
    # 1024x480 @ 85.6 Hz, 48 kHz hsync
    ModeLine "1024x480"    65.00 1024 1032 1176 1344   480  488  494
563 -hsync -vsync
    # 768x576 @ 79 Hz, 50 kHz hsync
    ModeLine "768x576" 50.00  768  832  846 1000  576  590  595  630
    # 768x576 @ 100 Hz, 61.6 kHz hsync
    ModeLine "768x576" 63.07  768  800  960 1024  576  578  590  616
EndSection

Section "Screen"
    Identifier "screen1"
    Device "Card0"
    Monitor "monitor1"
    DefaultColorDepth 16
    Subsection "Display"
        Depth 8
        Modes "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth 15
        Modes "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth 16
        Modes "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth 24
        Modes "800x600" "640x480"
    EndSubsection
EndSection

Section "ServerLayout"
    Identifier "XFree86 Configured"
    InputDevice "Mouse0" "CorePointer"
    InputDevice "Keyboard0" "CoreKeyboard"
    InputDevice "Keyboard1" "CoreKeyboard"
    Screen "screen1"
EndSection

Section "Files"
    ModulePath "/usr/X11R6/lib/modules"
    RgbPath "/usr/X11R6/lib/X11/rgb"
    FontPath "/usr/X11R6/lib/X11/fonts/misc/"
    #FontPath "/usr/X11R6/lib/X11/fonts/Speedo/"
    #FontPath "/usr/X11R6/lib/X11/fonts/Type1/"
    FontPath "/usr/X11R6/lib/X11/fonts/CID/"
    FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
```

```
    FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection

Section "Module"
    Load "GLcore"
    Load "glx"
    Load "extmod"
    #Load "dri"
EndSection

Section "InputDevice"
    Identifier "Keyboard0"
    Driver "keyboard"
    Option "XkbModel" "pc104"
    Option "XkbLayout" "us,ru,am"
    Option "XkbVariant" ",winkeys,"
    Option "XkbRules" "xfree86"
    Option "XkbOptions" "grp:alt_shift_toggle"
EndSection

Section "InputDevice"
    Identifier "Mouse0"
    Driver "mouse"
    Option "Protocol" "IMPS/2"
    Option "Device" "/dev/psaux"
    Option "ZAxisMapping" "4 5"
EndSection

Section "Device"
    Identifier "Card0"
    VendorName "Generic"
    BoardName "sis"
    #ChipSet "SIS300"
    Driver "sis"
    #Option "CHTVLumaBandwidthSVIDEO"    # <i>
    #Option "XvOnCRT2"             # [<bool>]
    #Option "BIOSFile"             # <str>
    Option "ColorHWCursorBlending" "True"
    #Option "ColorHWCursorBlendThreshold"# <i>
    #Option "CHTVOverscan"         # [<bool>]
    #Option "CHTVLumaFlickerFilter"      # <i>
    #Option "CHTVChromaFlickerFilter"    # <i>
    #Option "FastVram"             # [<bool>]
    #Option "TVXPosOffset"         # <i>
    #Option "SISTVSaturation"      # <i>
    #Option "TVYPosOffset"         # <i>
    #Option "SIS6326TVYFilterStrong"     # [<bool>]
    #Option "NoAccel"              # [<bool>]
    #Option "SIS6326TVEnableYFilter"     # [<bool>]
    #Option "NoXvideo"             # [<bool>]
    #Option "SISTVAntiFlicker"     # <i>
    #Option "Vesa"                 # [<bool>]
    #Option "ShadowFB"             # [<bool>]
    #Option "NoYV12"               # [<bool>]
```

```
    #Option "NoHostBus"            # [<bool>]
    #Option "RestoreBySetMode"     # [<bool>]
    #Option "CHTVContrast"         # <i>
    #Option "TurboQueue"           # [<bool>]
    #Option "CHTVChromaBandwidth"  # <i>
    #Option "HWcursor"             # [<bool>]
    #Option "SWcursor"             # [<bool>]
    #Option "MaxXFBMem"            # <i>
    #Option "CHTVCVBSColor"        # [<bool>]
    #Option "CHTVLumaBandwidthCVBS"      # <i>
    #Option "PanelDelayCompensation"     # <i>
    #Option "SISTVEdgeEnhance"     # <i>
    #Option "TVStandard"           # <str>
    #Option "ForceCRT1"            # [<bool>]
    #Option "DSTN"                 # [<bool>]
    #Option "CHTVSuperOverscan"    # [<bool>]
    #Option "NoInternalModes"      # [<bool>]
    #Option "UseROMData"           # [<bool>]
    #Option "ForceCRT2Type"        # [<str>]
    #Option "CHTVTextEnhance"      # <i>
    #Option "UseOEMData"           # [<bool>]
    #Option "SIS6326TVAntiFlicker" # <str>
    #Option "CHTVType"             # [<bool>]
    #Option "Rotate"               # [<str>]
    Option "UseColorHWCursor" "True"
EndSection

Section "DRI"
    mode 0666
EndSection

Section "ServerFlags"
EndSection
```

A.4 .xinitrc

```
#/home/nis/mural/

xhost +node140
if [ -f /usr/X11R6/bin/wmaker ]; then
      exec /usr/X11R6/bin/wmaker
else
      exec twm
fi
```

A.5 host.def

```
/* /home/nis/mural/dmx/xc/config/cf/   */
```

```
#define XFree86CustomVersion      "DMX Phase IV development code"
#define DefaultGcc2i386Opt        -O2
#define DefaultCCOptions         -ansi -pedantic GccWarningOptions \
                                  -Wno-redundant-decls -pipe
#define BuildServersOnly          YES
#define XdmxServer                YES
#define BuildDmxExt               YES
#define BuildDmxLibrary           YES
#define BuildXResLibrary          YES
#define BuildRenderLibrary        YES
#define XF86AFB                   NO
#define XnestServer               NO
#define XVirtualFramebufferServer NO
#define XprtServer                NO
#define BuildXIE                  NO
#define BuildPexExt               NO
#define SharedLibFont             NO
#define BuildXFree86ConfigTools   NO
#define BuildXF86DRI              YES
#define BuildXF86DRM              NO
#define XF86CardDrivers           sis vga
#define DriDrivers                sis
#define NormalLibGlx              NO
#define HasGlide3                 NO
```

A.6 dmx.conf

```
#/home/nis/mural/dmx/

virtual test {
      display node140:0 800x600 @0x0;
      display node141:0 800x600 @800x0;
      display node142:0 800x600 @0x600;
      display node143:0 800x600 @800x600;
}
```

A.7 .bashrc

```
#/home/nis/mural/

if [ -f /etc/bashrc ]; then
      . /etc/bashrc
fi
export PATH=$PATH:/home/nis/mural/cr-1.2/bin/Linux/
export LD_LIBRARY_PATH=/home/nis/mural/cr-1.2/lib/Linux/
xhost +node140
```

A.8 .crconfigs

```
#/home/nis/mural/

*/home/nis/mural/cr-1.2/mothership/configs/miautodmx.conf %m %p
```

A.9 .crsite

```
#/home/nis/mural/

{
    "mural_size" : (2, 2),         # (columnas, filas)
    "screen_size" : (800, 600),    # en píxeles
    "tile_size" : (800, 600),      # en píxeles
    "frontend_hosts" : ["node140"],
    "cluster_hosts" : ["node140", "node141", "node142", "node143"],
    "cluster_pattern" : ("node14#", 0)
}
```

A.10 miautodmx.conf

```
#/home/nis/mural/cr-1.2/mothership/configs/

crdir = "/home/nis/mural/cr-1.2/"
import random
import sys
sys.path.append( crdir + "/mothership/server" )
from mothership import *
if len(sys.argv) < 3:
    print "Chromium mothership error: no program specified to run!"
    sys.exit(1)

mothershipPort = int(sys.argv[1])
program = sys.argv[2]
cr = CR()
cr.MTU( 10*1024*1024 )
# Obtiene la información de configuración de .crsite
sf = 0
if os.environ.has_key("CRSITE"):
    siteFileName = os.environ["CRSITE"]
    try:
      sf = open(siteFileName, "r")
      print "Chromium miautodmx.conf: using $CRSITE"
    except IOError:
      sf = 0
if not sf:
    siteFileName = os.path.expanduser("~/.crsite")
    try:
```

```
      sf = open(siteFileName, "r")
      print "Chromium miautodmx.conf: using ~/.crsite"
    except IOError:
      sf = 0
if not sf:
    siteFileName = os.path.expanduser("/etc/crsite")
    try:
      sf = open(siteFileName, "r")
      print "Chromium miautodmx.conf: using /etc/crsite"
    except IOError:
      sf = 0
if not sf:
    print "Chromium mothership error: unable to open ~/.crsite
file!"
    sys.exit(1)
sf.close()
# Obtiene el tamaño y los hostnames para DMX
TILE_COLS = 2 #siteInfo["mural_size"][0]
TILE_ROWS = 2 #siteInfo["mural_size"][1]
HOSTS = ['node140','node141','node142','node143']
#siteInfo["cluster_hosts"]
localHostname = os.uname()[1]
# Escoge un puerto aleatorio para la comunicación con el servidor
serverPort = random.randint(7000, 7100)
AUTOSTART = 1
# N
TILE_WIDTH = 550
TILE_HEIGHT = 550
tilesortspu = SPU('tilesort')
tilesortspu.Conf('use_dmx', 1)
tilesortspu.Conf('retile_on_resize', 1)  # the default
tilesortspu.Conf('bucket_mode', 'Non-Uniform Grid')
clientnode = CRApplicationNode( )
#clientnode.StartDir( crbindir )
clientnode.SetApplication( program )
clientnode.AddSPU( tilesortspu )
clientnode.Conf('track_window_size', 1)
for row in range(TILE_ROWS):
    for col in range(TILE_COLS):
      n = row * TILE_COLS + col
      print n
      print HOSTS[n]
      renderspu = SPU( 'render' )
      renderspu.Conf('try_direct',1)
      renderspu.Conf('force_direct',0)
      renderspu.Conf('display_string', HOSTS[n] + ":0")
      renderspu.Conf('render_to_app_window', 1)
      renderspu.Conf( 'window_geometry', [1.1*col*TILE_WIDTH,
1.1*row*TILE_HEIGHT, TILE_WIDTH, TILE_HEIGHT] )
      renderspu.Conf('system_gl_path','/usr/X11R6/lib/')
      servernode = CRNetworkNode( HOSTS[n] )
      servernode.AddTile( col*TILE_WIDTH, (TILE_ROWS-row-
1)*TILE_HEIGHT, TILE_WIDTH, TILE_HEIGHT )
      servernode.AddSPU( renderspu )
```

```
      servernode.Conf('optimize_bucket', 0)
      servernode.Conf('use_dmx', 1)
      cr.AddNode( servernode )
      if AUTOSTART:
          if not n:
            servernode.AutoStart( '/home/nis/mural/cr-
1.2/bin/Linux/crserver -mothership node140:%d' % (mothershipPort) )
          if n:
            servernode.AutoStart( ["/usr/bin/rsh", HOSTS[n],
            "/bin/sh -c 'DISPLAY=:0.0 LD_LIBRARY_PATH=/usr/lib
/home/nis/mural/cr-1.2/bin/Linux/crserver -mothership %s:%d'" %
(localHostname, mothershipPort) ] )
          tilesortspu.AddServer( servernode, protocol='tcpip',
port=serverPort )
cr.AddNode(clientnode)
cr.Go( mothershipPort )
```

B APÉNDICE B: DATOS DE LAS PRUEBAS DE RENDIMIENTO

B.1 Rendimiento (en cuadros por segundo) de un sistema distribuido con DMX y Chromium

	Atlantis	city	glxgears
	64.2	2.7	48.1
	64.7	3.0	48.1
	67.3	2.8	48.1
	54.3	2.4	41.4
	59.6	2.7	48.1
	65.3	2.9	48.1
	67.3	2.5	48.1
	46.8	2.7	48.1
	65.5	3.1	48.1
	58.7	2.7	48.1
	60.2	2.6	48.1
	62.6	2.7	41.4
	62.4	2.8	48.1
	65.0	2.5	48.1
	63.9	2.6	48.1
	40.6	2.8	48.1
	55.8	2.6	48.1
	57.9	2.4	48.1
	60.9	2.8	41.4
	51.5	2.7	48.1
Promedio	**59.7**	**2.7**	**47.1**

B.2 Rendimiento (en cuadros por segundo) de un solo computador

	Atlantis	city	glxgears
	80.5	3.2	60.0
	80.4	3.3	60.0
	82.3	3.2	50.0
	74.3	3.1	60.0
	74.6	3.5	60.0
	81.5	3.4	50.0
	81.0	3.4	60.0
	80.1	3.2	60.0
	80.7	3.3	60.0
	82.5	3.2	50.0
	75.2	3.3	60.0
	75.6	3.5	60.0
	80.1	3.3	60.0
	80.8	3.4	66.7
	80.7	3.2	60.0
	80.9	3.1	60.0
	80.5	3.3	60.0
	80.3	3.3	66.7
	82.3	3.5	60.0
	75.2	3.3	60.0
Promedio	**79.5**	**3.3**	**59.2**

B.3 Rendimiento (en cuadros por segundo) de un sistema distribuido con DMX y sin Chromium

	Atlantis	city	glxgears
	24.0	0.8	20.0
	23.9	0.8	20.0
	24.0	0.8	22.2
	23.4	0.9	20.0
	23.8	0.8	22.0
	21.0	0.8	20.0
	23.1	0.9	22.2
	23.4	0.8	22.2
	22.9	0.8	20.0
	21.5	0.9	20.0
	21.5	0.8	20.0
	20.9	0.9	20.0
	19.6	0.8	22.2
	18.4	0.7	22.2
	24.2	0.8	20.0
	24.3	0.9	22.2
	24.4	0.8	20.0
	24.5	0.8	22.2
	24.1	0.8	20.0
	23.8	0.8	20.0
Promedio	**22.8**	**0.8**	**20.9**

C APÉNDICE C: DATOS DE LATITUD, LONGITUD ALTURA Y PLUVIOSIDAD EN LA COSTA ECUATORIANA ENTRE ENERO DE 1982 Y DICIEMBRE DE 1983

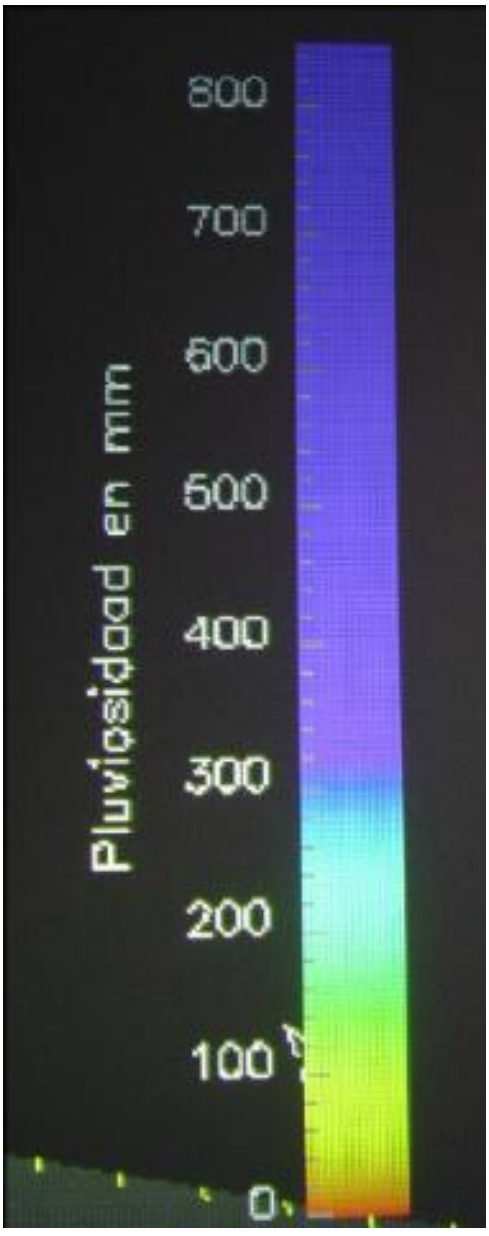
C.1 Latitud, longitud y altura

	Latitud	Longitud	Altura (m)
Bahía de Caráquez	-0.57	-80.40	3.00
Esmeraldas	0.97	-79.62	7.00
Guayaquil	-2.15	-79.88	5.00
Manta	-0.95	-80.68	11.90
Portoviejo	-0.14	-80.47	43.90

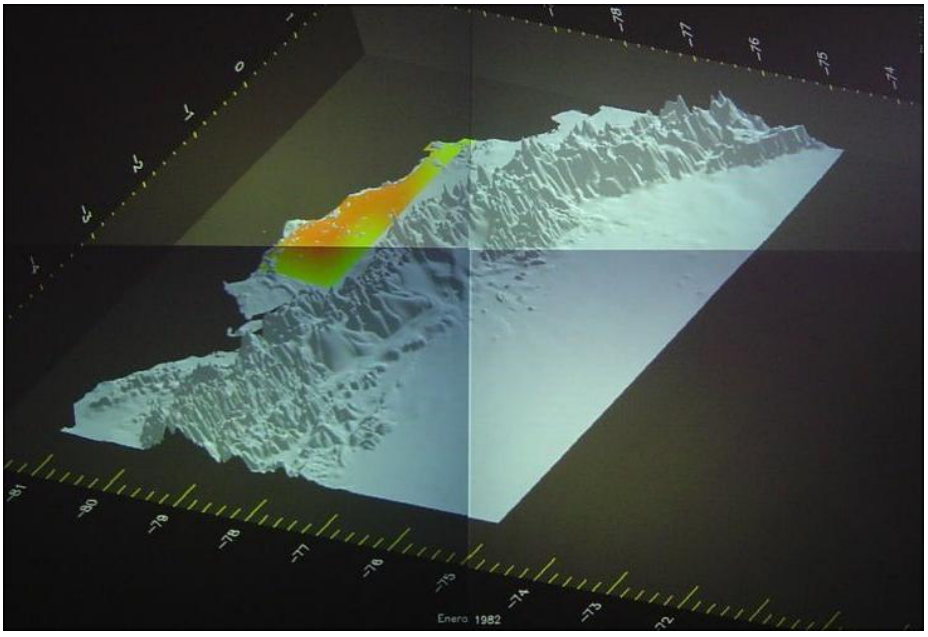
C.2 Pluviosidad (en milímetros)

	Bahía de Caráquez		Esmeraldas		Guayaquil		Manta		Portoviejo	
	1982	**1983**	**1982**	**1983**	**1982**	**1983**	**1982**	**1983**	**1982**	**1983**
Ene	17.90	344.80	99.90	225.70	126.80	601.70	10.10	265.80	9.70	230.30
Feb	146.50	236.60	120.70	358.00	89.80	539.40	5.40	119.30	6.20	45.30
Mar	18.40	323.60	146.50	198.60	6.70	830.50	0.00	152.80	9.50	56.60
Abr	37.70	308.90	78.10	120.30	16.00	606.40	0.70	338.40	1.00	176.60
May	8.10	594.80	63.20	96.00	14.10	621.70	4.80	436.20	0.30	259.10
Jun	1.50	571.10	15.10	149.70	0.00	629.90	0.00	151.70	2.30	342.10
Jul	0.00	467.50	30.30	128.70	0.00	292.50	0.00	267.80	0.00	189.00
Ago	0.00	19.90	7.30	48.70	0.00	18.20	0.00	9.10	0.00	25.10
Sep	6.50	128.70	1.80	52.90	0.60	18.90	0.00	20.00	1.00	42.40
Oct	35.80	1.40	15.90	151.70	6.30	4.00	11.80	0.10	26.20	0.40
Nov	119.60	7.90	127.20	79.50	152.30	1.10	23.30	1.70	61.00	0.60
Dic	37.60	22.80	99.20	26.70	255.50	66.40	38.90	18.90	92.00	22.50

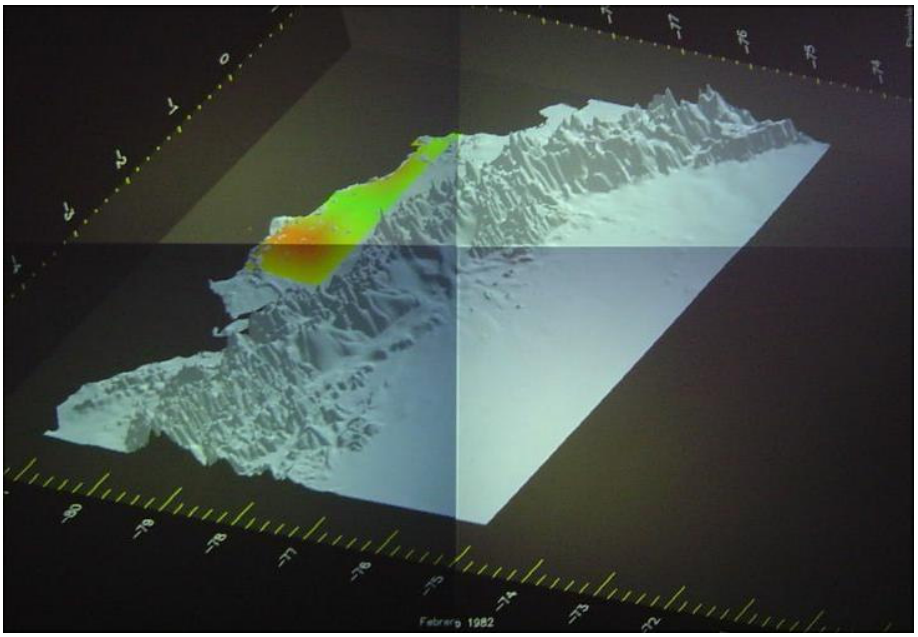
D APÉNDICE D: RESULTADOS DE PLUVIOSIDAD EN LA COSTA ECUATORIANA ENTRE ENERO DE 1982 Y DICIEMBRE DE 1983



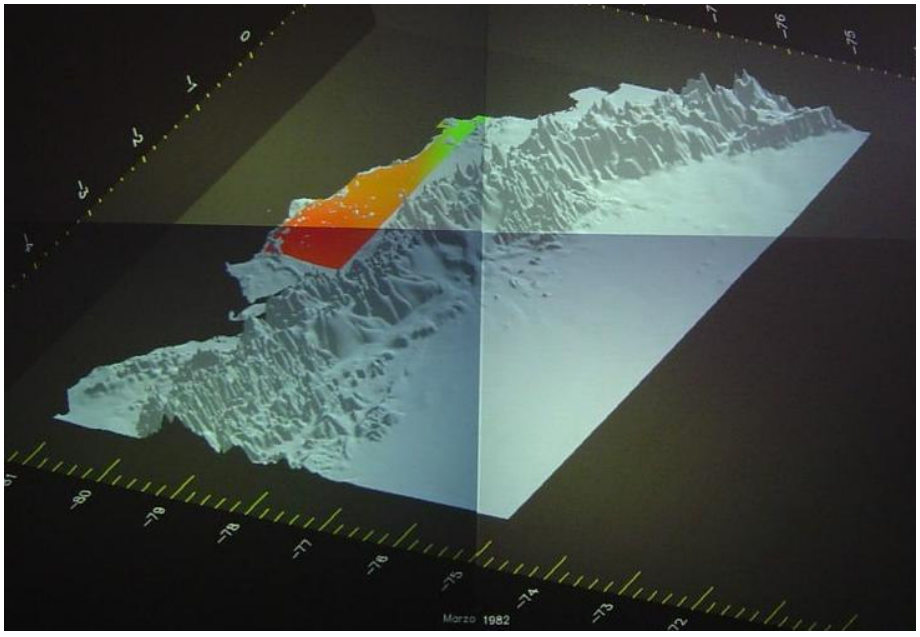
Guía de colores para los niveles de pluviosidad



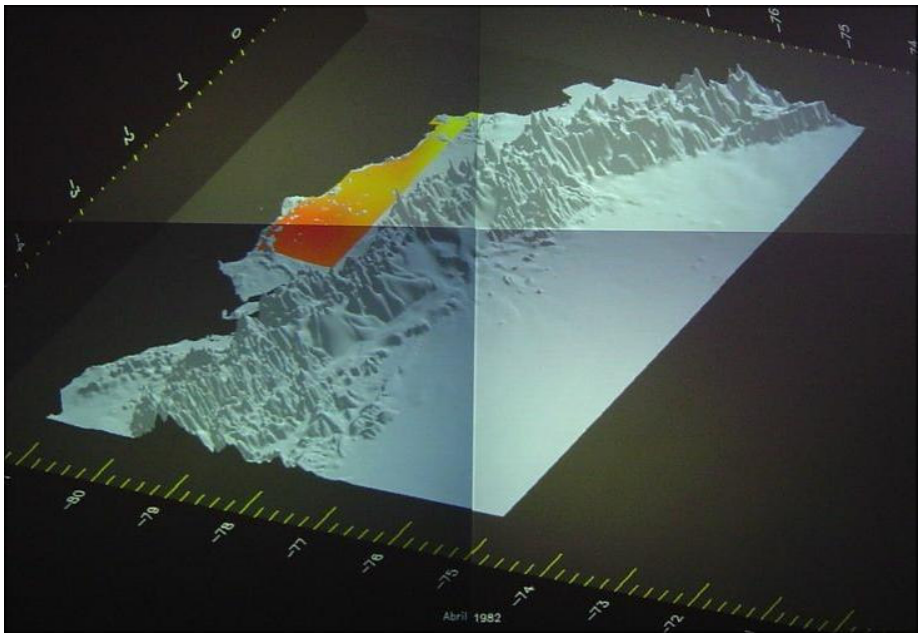
Enero 1982



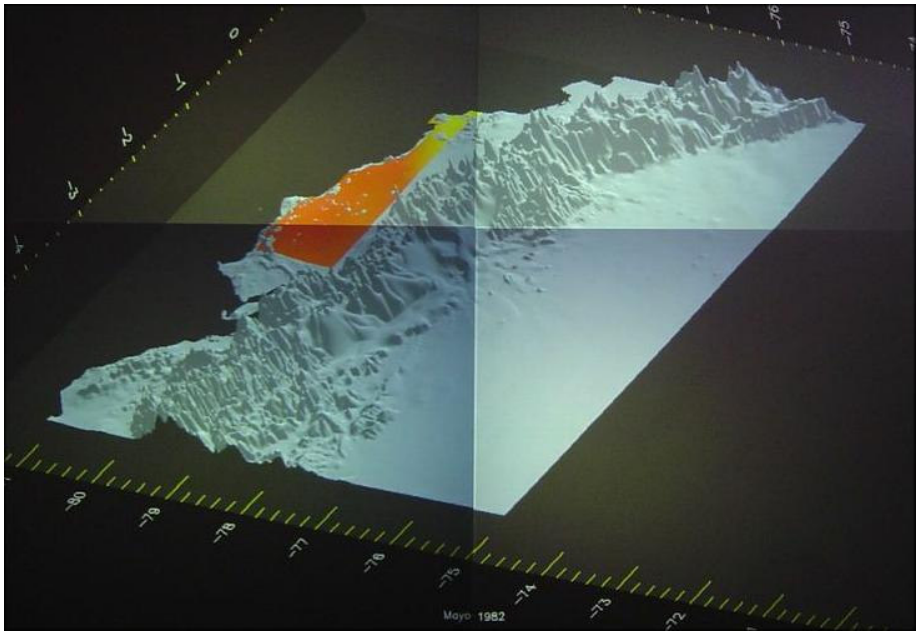
Febrero 1982



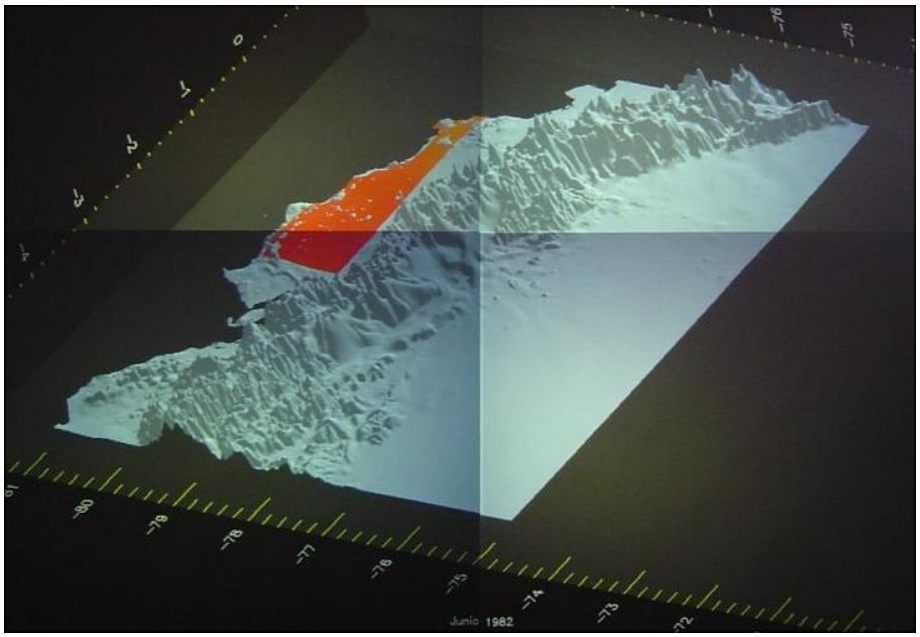
Marzo 1982



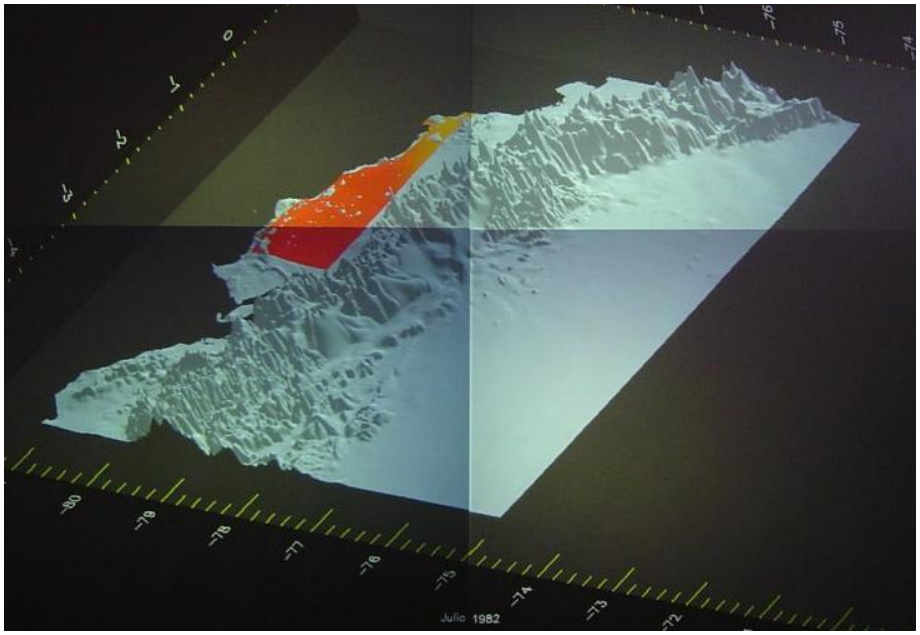
Abril 1982



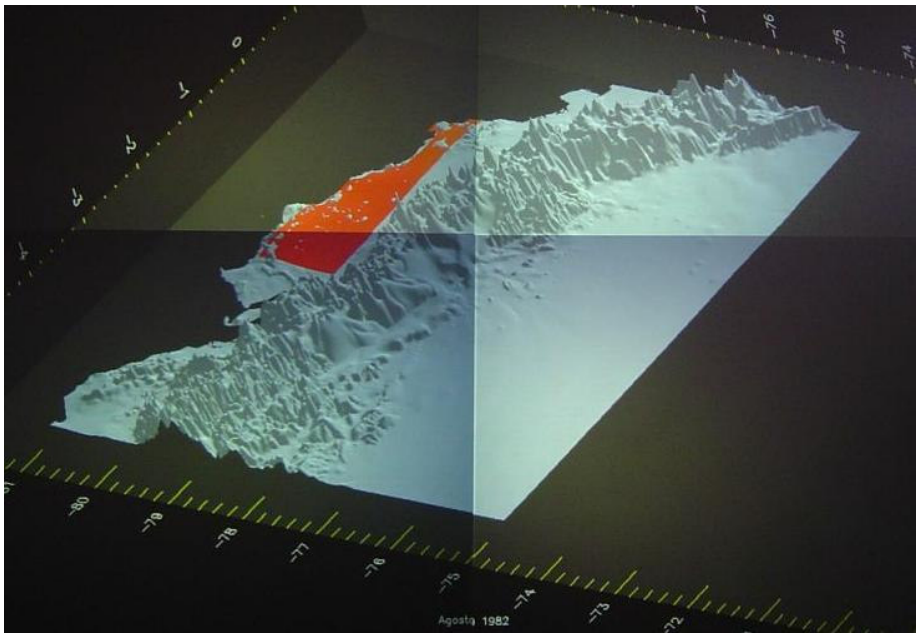
Mayo 1982



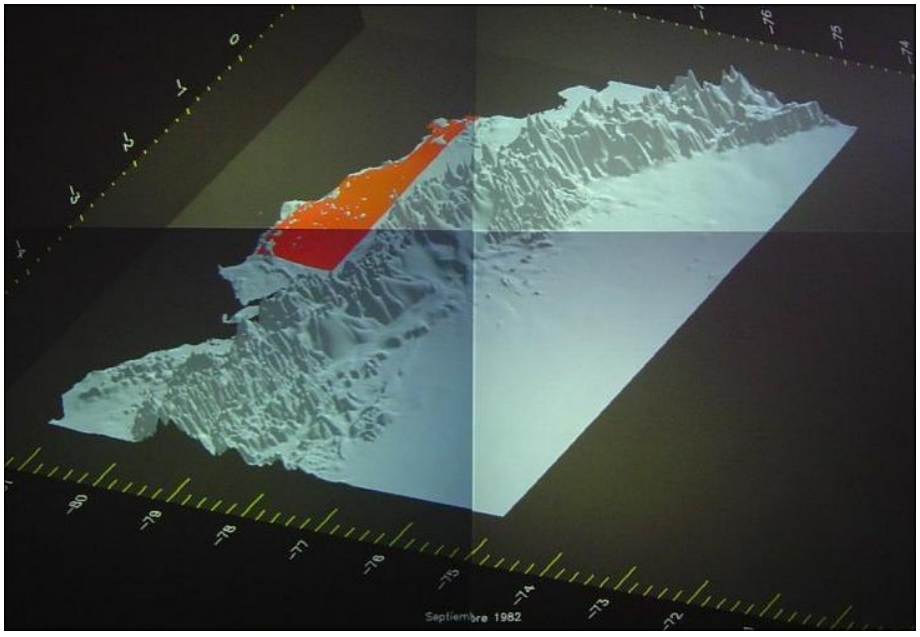
Junio 1982



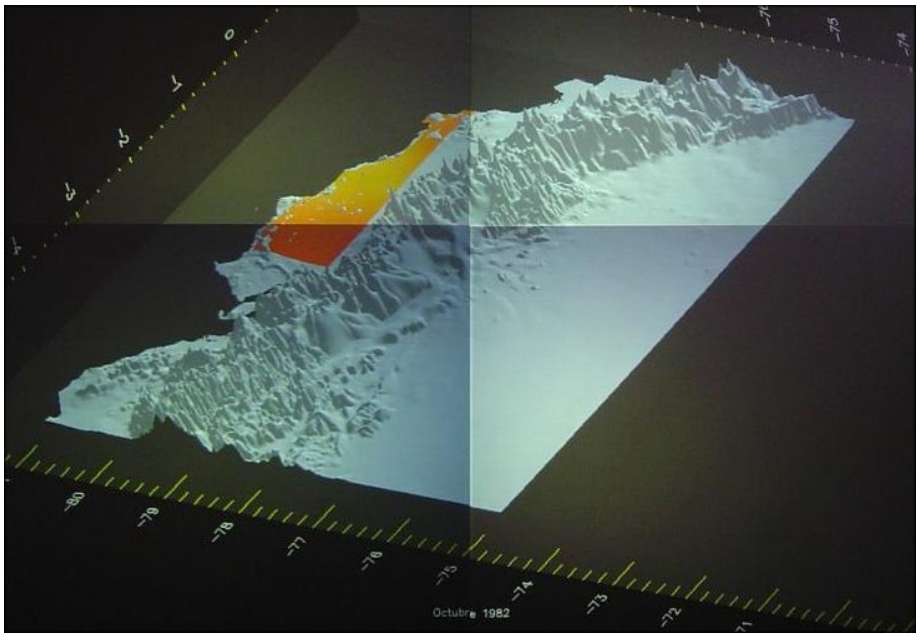
Julio 1982



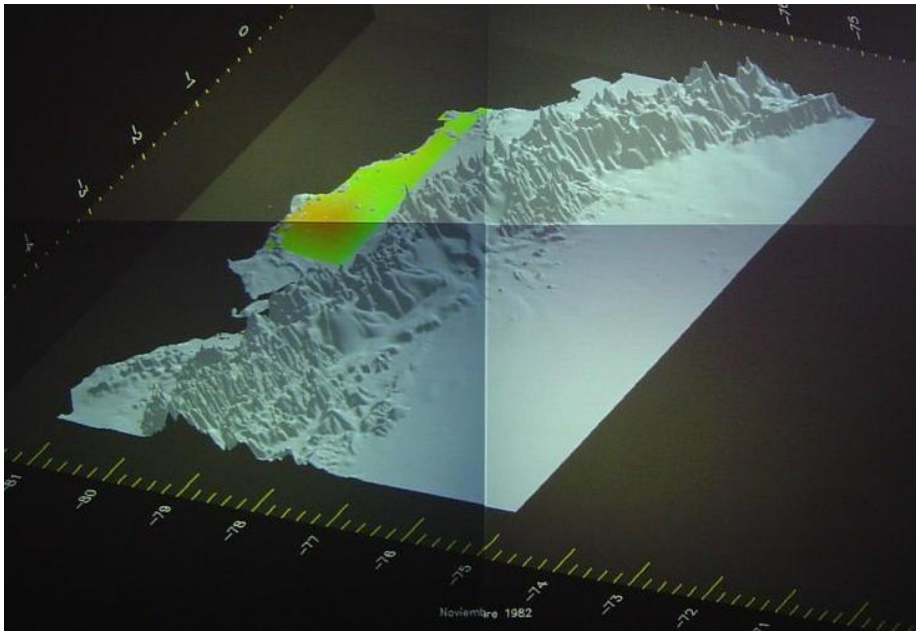
Agosto 1982



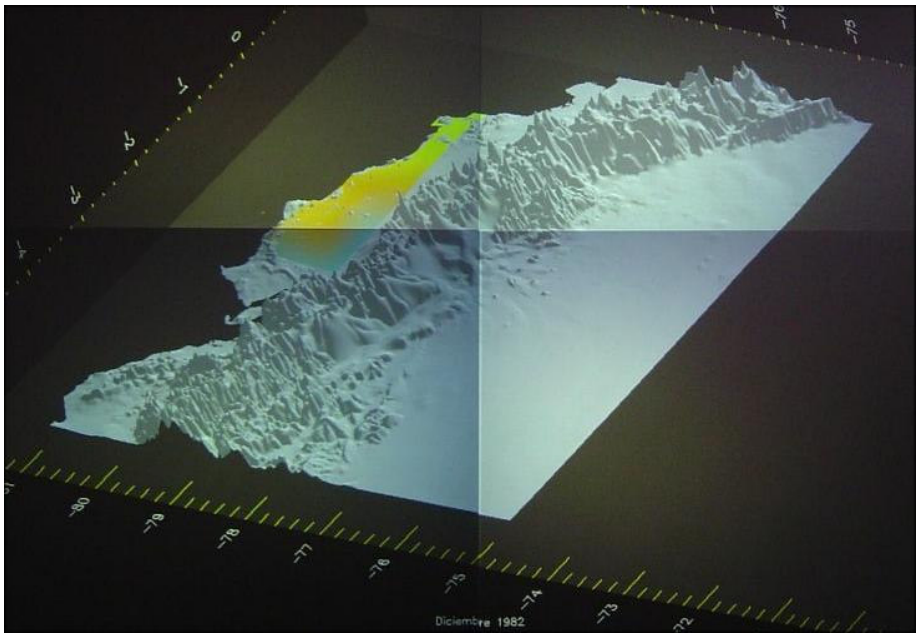
Septiembre 1982



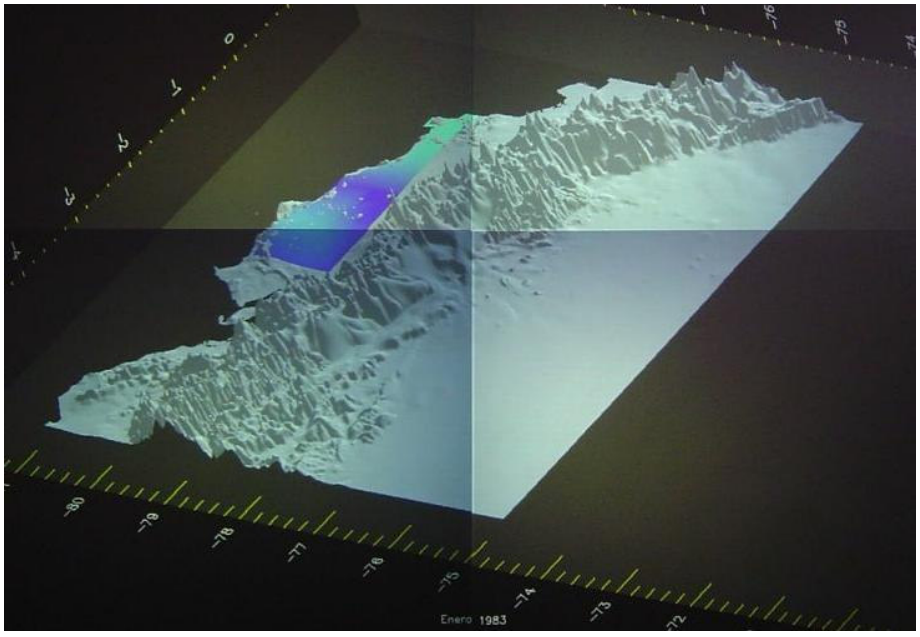
Octubre 1982



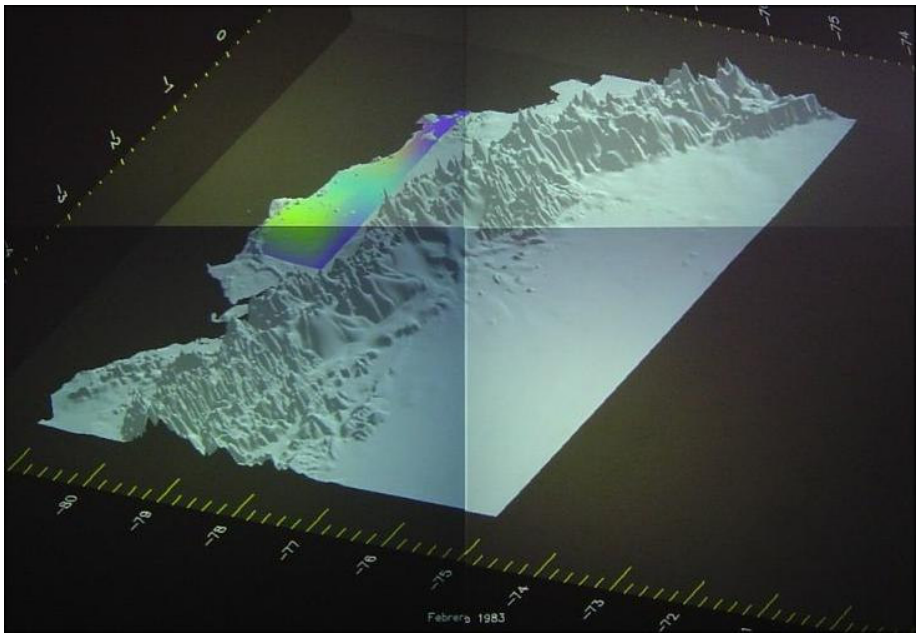
Noviembre 1982



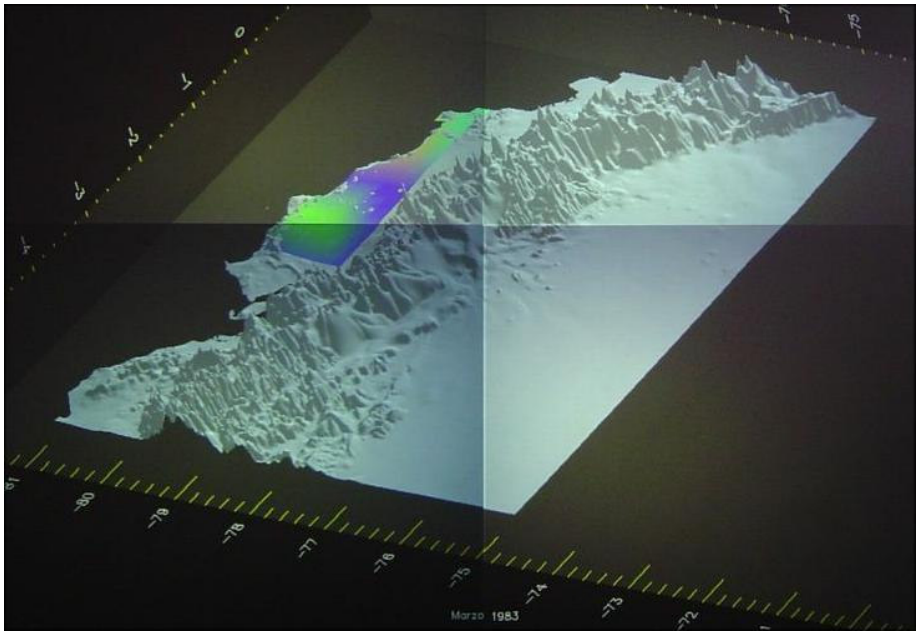
Diciembre 1982



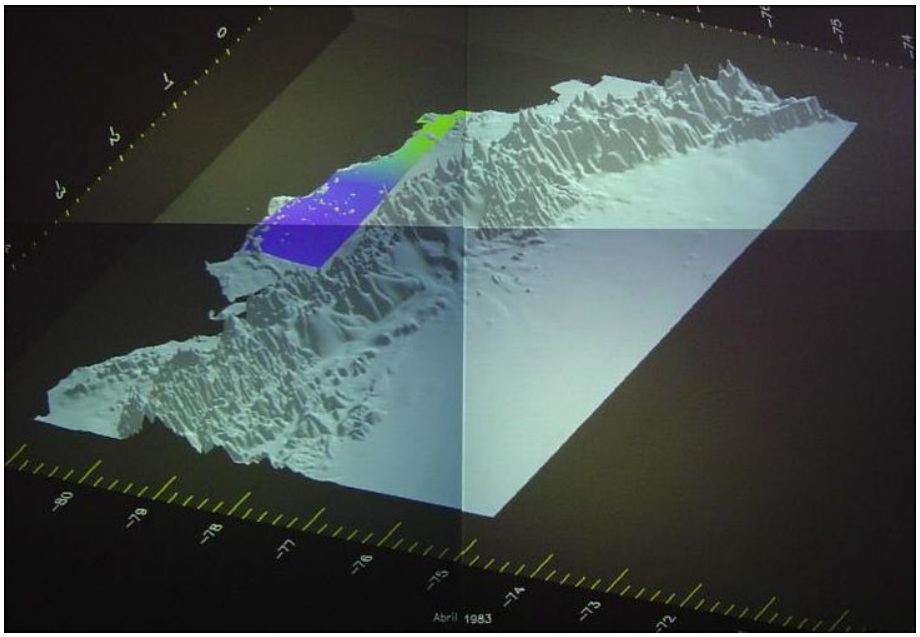
Enero 1983



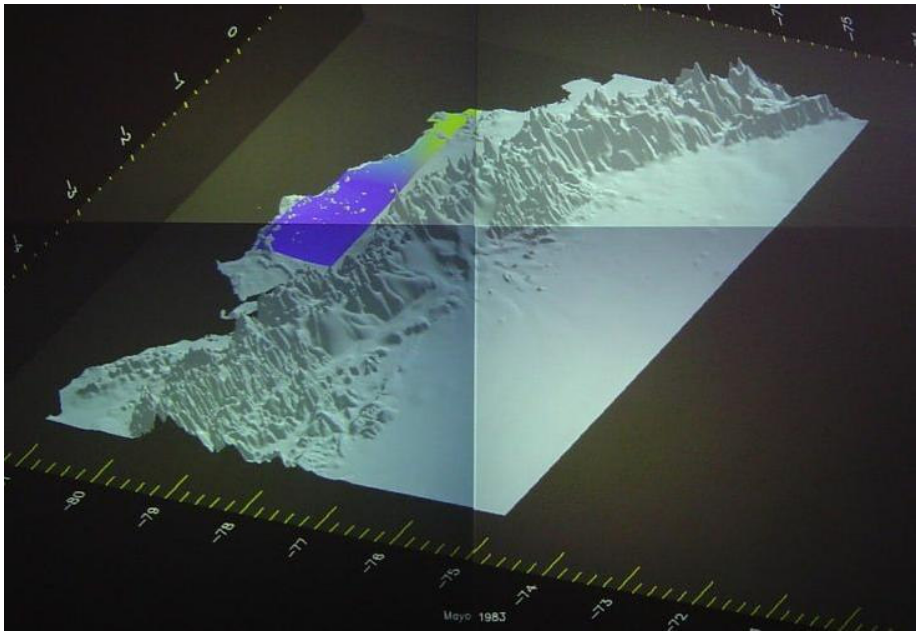
Febrero 1983



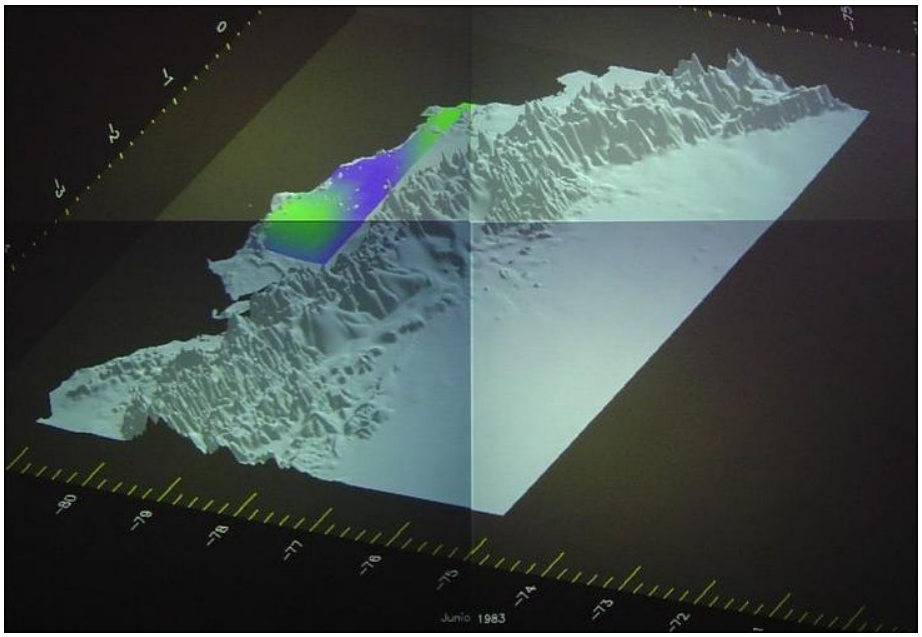
Marzo 1983



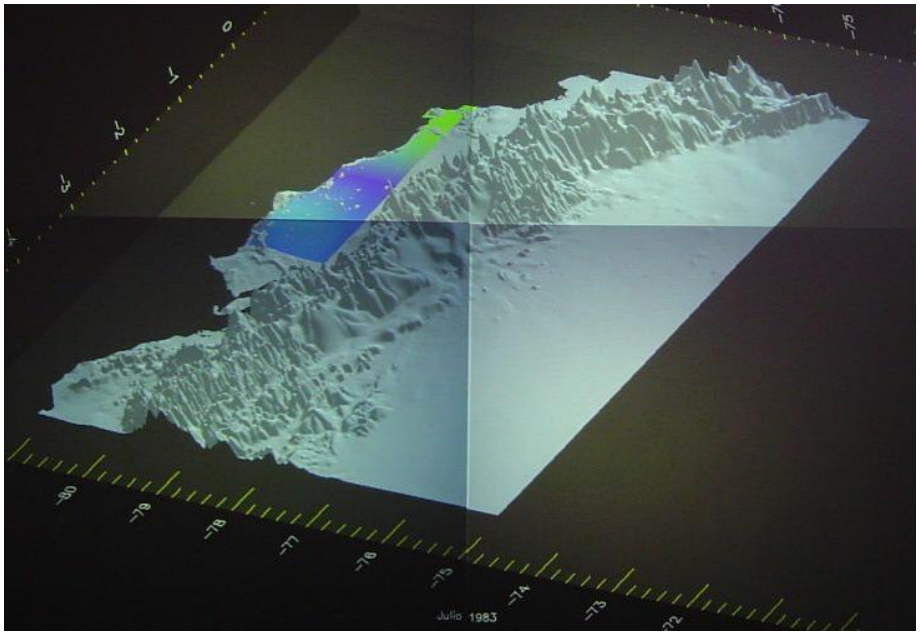
Abril 1983



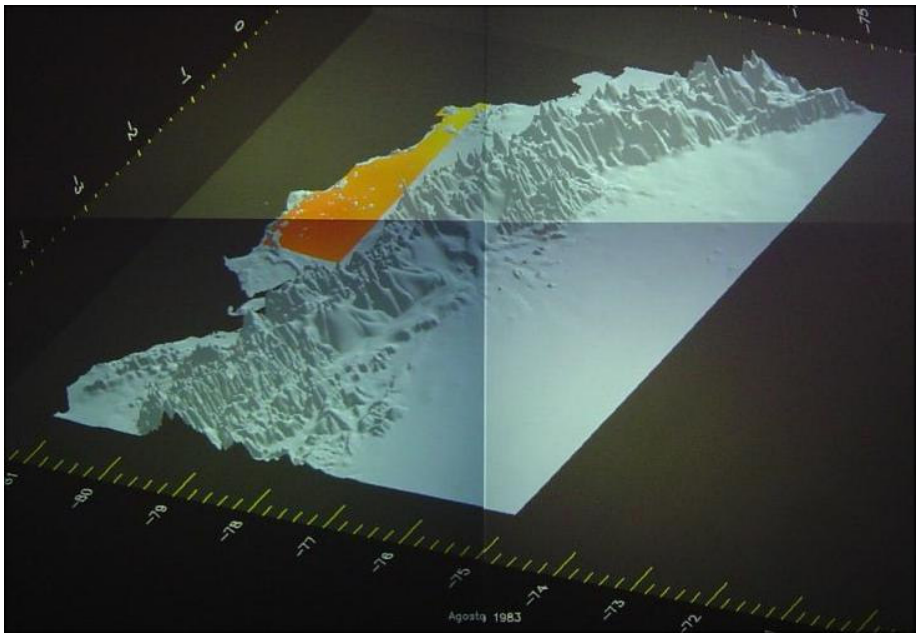
Mayo 1983



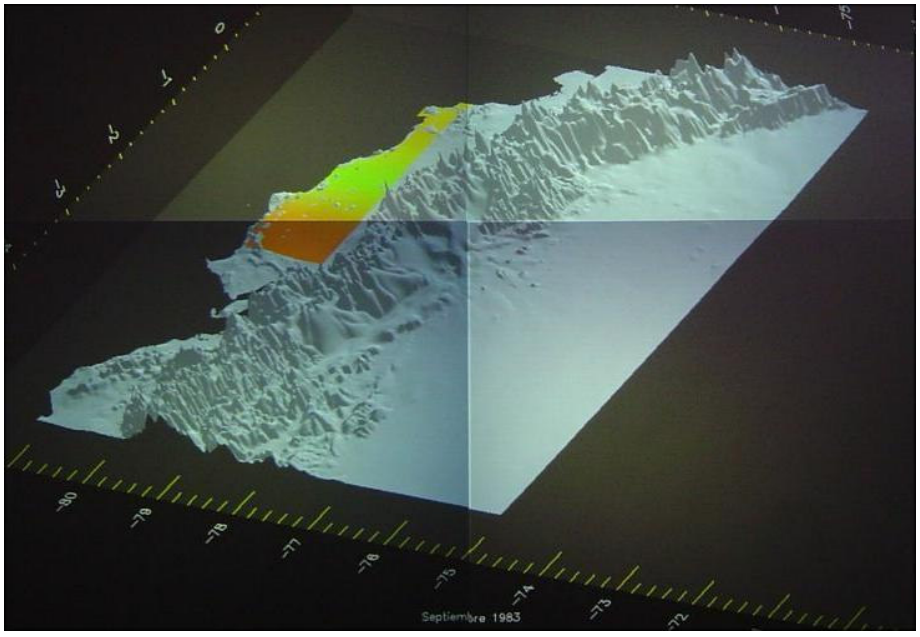
Junio 1983



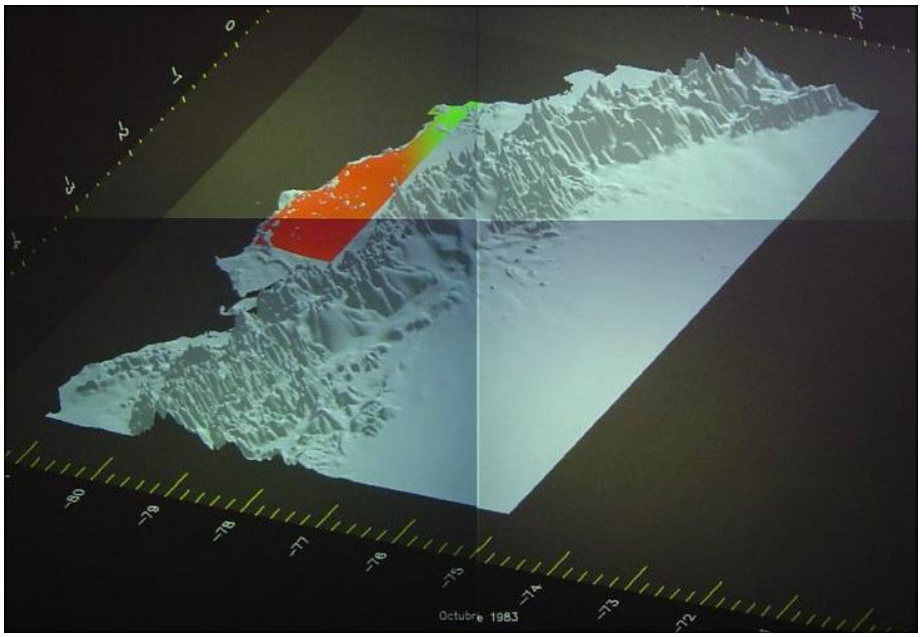
Julio 1983



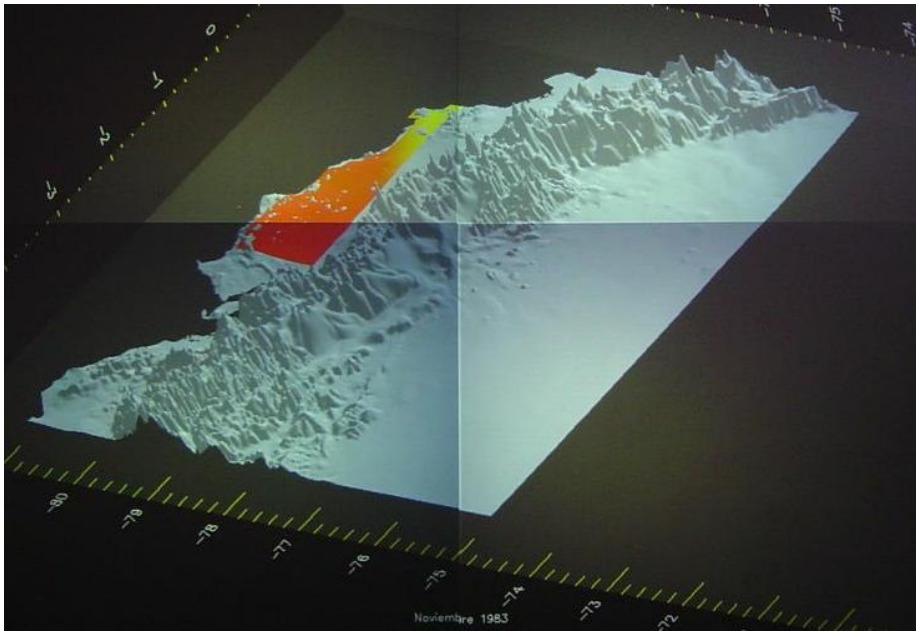
Agosto 1983



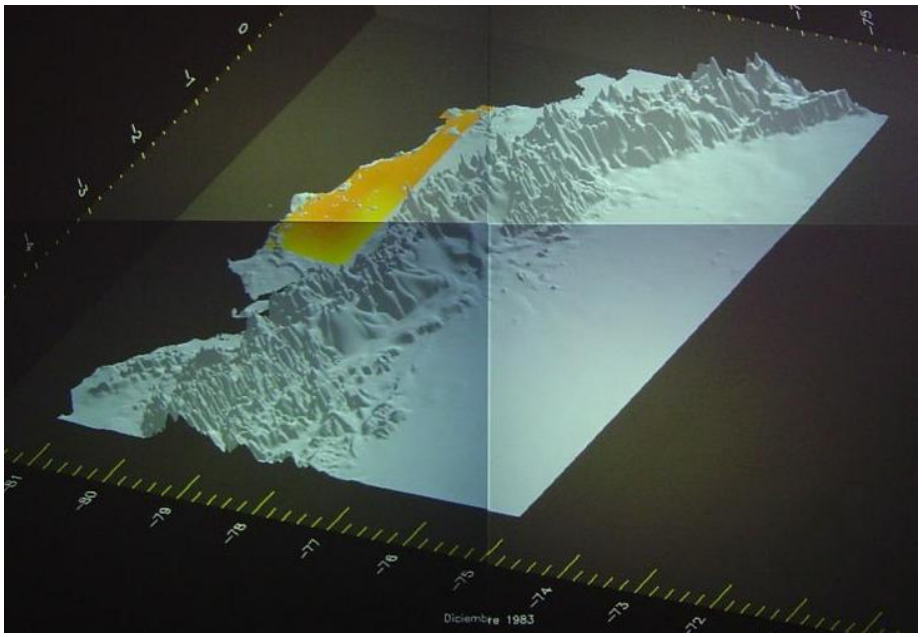
Septiembre 1983



Octubre 1983



Noviembre 1983



Diciembre 1983

E APÉNDICE E: PRESUPUESTO DE LOS EQUIPOS NECESARIOS PARA CONSTRUIR UN SISTEMA DE VISUALIZACIÓN DE ALTA RESOLUCIÓN

Accesorio	Cantidad	Valor Unitario	Valor Total
Computadores	4	$ 499.89	$ 1,999.56
Proyectores de 1,500 lúmenes	4	$ 1,690.08	$ 6,760.32
Switch	1	$ 37.50	$ 37.50
Teclado y ratón inalámbrico	1	$ 35.00	$ 35.00
Pantalla de proyección trasera	1	$ 1,060.00	$ 1,060.00
Base pantalla	1	$ 155.00	$ 155.00
Mueble y base de los proyectores	1	$ 535.00	$ 535.00
		Total	$ 10,047.38

REFERENCIAS

[1] Alonso, Juan J. y Maximiliano Fatica. “Parallel Methods in Numerical Analysis Course Info.” Abril 26, 2004. <http://redhot.stanford.edu/cs238/WWW/index.html>.

[2] “Anti-Aliasing.” Enero 6, 2004. <http://www.widearea.co.uk/designer/anti.html>.

[3] Arnaiz, Jesús. "Supercomputación y proceso en paralelo." Enero 27, 2003. <http://html.rincondelvago.com/supercomputacion-y-proceso-en-paralelo.html>.

[4] Barlow, Kevin. “What is a Lumen?” Technologies for Worship Magazine Online. Marzo / Abril 2000.

[5] Baum, Steven. “Linux Software Encyclopedia.” Texas A&M University. Marzo, 2001.

[6] “BenchADM (40-cubed).” Abril 27, 2004.
<http://www.cactuscode.org/Benchmark/BenchADM40/BenchADM40_Results.html>.

[7] “Beowulf Software.” Octubre 3, 2003.
<http://www.beowulf.org/software/software.html>.

[8] Calle, Dan. “Supercomputers.” Octubre 3, 2003.
<http://ei.cs.vt.edu/~history/SUPERCOM.Calle.html>.

[9] “Chromium Homepage.” Octubre 3, 2003.
<http://chromium.sourceforge.net>.

[10] “Computer Online.” Abril 27, 2004.
<http://www.gotocol.com/sgioctane.html>.

[11] “ComputerUser.com High-Tech Dictionary.” Octubre 1, 2003.
<http://www.computeruser.com/resources/dictionary/index.html>.

[12] Des Ligneris, Benoît et al. “Open Source Cluster Application Resources (OSCAR): design, implementation and interest for the [computer] scientific community.” OSCAR Symposium, Sherbrooke. Mayo 2003.

[13] “Digital Photo Reference.” Abril 15, 2004. <http://pixlabs.com/Support/phototutorial.htm>.

[14] “DirectX.org.” Octubre 3, 2003. <http://www.directx.org>.

[15] “DMX Homepage.” Octubre 3, 2003. <http://dmx.sourceforge.net>.

[16] Dongarra, Jack y Aad J. van der Steen. "Overview of Recent Supercomputers.” The Hague, NCF, 7ma. Edición, 1997.

[17] “Hammer Imports: Projectors.” Enero 6, 2004. <http://www.hammerimports.com/Merchant2/merchant.mv?Screen=CTGY&Store_Code=HI&Category_Code=02-PJ>.

[18] “HowStuffWorks - Learn how Everything Works!” Octubre 3, 2003. <http://www.howstuffworks.com>.

[19] Hsieh, Jenwei. "High-Performance Computing with Beowulf Clusters." Dell Power Solutions, Issue 2, 2000.

[20] Humphreys, Greg et al. "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters." SIGGRAPH 2002.

[21] Humphreys, Greg et al. "WireGL: A Scalable Graphics System for Clusters." SIGGRAPH 2001.

[22] "IBM Research Visualization Data Explorer." Febrero 16, 2004. <http://www.research.ibm.com/dx/>.

[23] "Introduction to Parallel Computing." Abril 21, 2004. <http://www.llnl.gov/computing/tutorials/parallel_comp/#DesignPartitioning>.

[24] Lemerie, Jan. "Parallel Systems Course Info." Enero 27, 2003. <http://parallel.vub.ac.be>.

[25] "Manchester Visualization Centre." Julio 31, 2004. <http://www.sve.man.ac.uk/mvc/>.

[26] "MandrakeSoft - CLIC Clustering project." Octubre 3, 2003. <http://clic.mandrakesoft.com/index-en.html>.

[27] Martin, Kevin E. et al. "Distributed Multihead X design." Red Hat, Inc., Durham, North Carolina. Julio 2003.

[28] Milliken, Jr., M. K. "Differences between Front and Rear Projection." Angles of View. Vol I, No. 4. Mayo 1995.

[29] Mock, Kenrick. "Introduction to Computer Architecture Lecture Notes." Abril 25, 2004. <http://www.math.uaa.alaska.edu/~afkjm/cs221/>.

[30] Morgan, Timothy Prickett. "SGI Debuts Onyx Visualization Server, Tezro Workstations." Breaking News. Julio 14, 2003.

[31] Mueller, Klaus. "Introduction to Visualization." Abril 25, 2004. <http://www.cs.sunysb.edu/~mueller/teaching/cse332/intro.pdf>.

[32] "NCSA VR Lab Home Page." Julio 31, 2004. <http://www.ncsa.uiuc.edu/VR/VR/VRHomePage.html>.

[33] Nilssen, Andy y Andrew D. Davis. "TANDBERG's Native Resolution What It Does and Why It's the Clear Solution." Wainhouse Research. Septiembre 2001

[34] "Open Visualization Data Explorer." Octubre 3, 2003. <http://www.opendx.org/>.

[35] "OpenGL - High Performance 2D/3D Graphics." Octubre 3, 2003. <http://www.opengl.org>.

[36] "OpenGL vs. DirectX." Abril 16, 2004. <http://www.csis.gvsu.edu/~sowersa/adv.html>.

[37] Orcero, David Santo. "Simulación y optimización paralela de agregados del Silicio." Disertación de Maestría en Ciencias de la Computación. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga. Mayo 2002.

[38] "OSCAR: Open Source Cluster Application Resources." Octubre 3, 2003. <http://oscar.sourceforge.net>.

[39] "Parallel Computing Introduction." The National Center for Supercomputing Applications. Abril 27, 2004. <http://archive.ncsa.uiuc.edu/SCD/Training/materials/html/intro/>.

[40] "Projector Point." Octubre 2, 2003. <http://www.projectorpoint.co.uk>.

[41] "Projector questions answered." Multimedia Projectors. Octubre 2, 2003. <http://www.multimediaprojectors.co.uk/faqs.htm>.

[42] "Projector Resolution - One Factor in Picture Quality." Projector Central. Octubre 2, 2003. <http://www.projectorcentral.com/buyers2.htm>.

[43] "ProjectorCentral: Compare Projectors." Projector Central. Octubre 2, 2003. <http://www.projectorcentral.com/parts_compare.cfm?pid_1=1819&pid_2=1478&pid_3=1332&pid_4=1333>.

[44] "RealVNC." Febrero 12, 2004. <http://www.realvnc.com/>.

[45] "RemacoTech.com." Octubre 2, 2003. <http://www.remacotech.com/07.faq.htm>.

[46] Richarson, Tristan et al. "Virtual Network Computing." IEEE Internet Computing. Vol.2 No.1. Enero / Febrero 1998.

[47] "RIEGNER." Octubre 2, 2003. <http://www.riegner.com.co/capacitacion.htm>.

[48] "Running X11 - 1 Introduction." Abril 27, 2004. <http://fink.sourceforge.net/doc/x11/intro.php?phpLang=en>.

[49] "Scientific Visualization Studio (SVS)." Julio 31, 2004. <http://svs.gsfc.nasa.gov/>.

[50] "SETI@home: Search for Extraterrestrial Intelligence at home." Abril 21, 2004. <http://setiathome.ssl.berkeley.edu>.

[51] Shankland, Stephen. "InfiniBand reborn for supercomputing." Abril 26, 2004. <http://news.com.com/2100-1001-966777.html>.

[52] "Silicon Graphics Incorporated." Abril 27, 2004. <http://www.sgi.com/>.

[53] "UCLA Sandbox Project: Tiled Display Pilot Project." Febrero 17, 2004. <http://www.ats.ucla.edu/at/tiled_displays/software_and_status.htm>.

[54] "US EPA Sci Vis Center - What is Visualization?" Octubre 23, 2003. <http://www.epa.gov/vislab/svc/overview/what_is_vis.html>.

[55] "VGEE Introduction." Abril 15, 2004. <http://www.dpc.ucar.edu/vgee/intro.htm>.

[56] "Vis5d+ - Home Page." Octubre 3, 2003. <http://vis5d.sourceforge.net>.

[57] "Visualization." Octubre 23, 2003. <http://www.eastnet.ecu.edu/revitalise/visualization.htm>.

[58] "Visualization Lab." Julio 31, 2004. <http://www.cs.sunysb.edu/~vislab/vislab_home.html>.

[59] "VNC." Octubre 3, 2003. <http://www.ncsa.uiuc.edu/TechFocus/Deployment/DBox/Doc/vnc.html>.

[60] "VNC - Virtual Network Computing from AT&T Laboratories Cambridge." Febrero 18, 2004. <http://www.uk.research.att.com/archive/vnc/index.html>.

[61] Wallace, Grant et al. “DeskAlign: Automatically Aligning a Tiled Windows Desktop.“ PROCAMS. Octubre 2003.

[62] “Webopedia: Online Dictionary for Computer and Internet Terms.” Octubre 1, 2003. <http://www.webopedia.com>.

[63] “WhatIs.com.” Octubre 1, 2003. <http://whatis.techtarget.com>.

[64] “WireGL: Software for OpenGL Tiled Rendering.” Octubre 3, 2003. <http://www-graphics.stanford.edu/software/wiregl>.

[65] “WireGL and Chromium.” Abril 27, 2004. <http://oldsite.vislab.usyd.edu.au/research/display_soft/chrom-wgl/main/main.html>.

[66] Wright Jr., Richard S. y Michael Sweet. “OpenGL Super Bible!” Wait Group Press. 1996.

[67] Yaschenco, Andy. “CRT, LCD, OLED: Evolution of the Screen...” XbitLabs Articles. Mayo, 2003.