UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS ARAGÓN

"VISUALIZACIÓN GRÁFICA DE PARÁMETROS DE CONTAMINACIÓN EN POZOS Y DEPÓSITOS DE AGUA EN CIUDAD NEZAHUALCÓYOTL"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A N:

ALEJANDRO VILLAGÓMEZ DÍAZ

FABIÁN ALFONSO RODRÍGUEZ MARTÍNEZ

ASESOR DE TESIS:
M. EN C. MARCELO PÉREZ MEDEL

MÉXICO, D.F. **2003**

AGRADECIMIENTOS

A Dios por no dejarnos desfallecer e iluminarnos en los momentos mas difíciles, permitiendo que mis seres queridos estén conmigo en estos momentos.

Con todo cariño a mi Mamá, a quien le debo todo lo que soy, gracias a todos sus esfuerzos y sacrificios estoy aquí ahora.

A la memoria de mi Padre que aun que no estas conmigo fuiste un ejemplo a seguir.

A Fabi por tu apoyo, cariño y comprensión que a pesar de todas la adversidades que hemos pasado, juntos las superamos. Gracias por ser mi hermana y estar aquí.

A ti Chiquita por traer a mi vida, alegrías, ilusiones, por estar conmigo en un momento muy importante de mi vida, por tu apoyo incondicional, por ser mi ilusión.

A todos aquellos que formaron parte de este proyecto en especial a el Ing. Marcelo Pérez, mil gracias.

Fabián Alfonso Rodríguez Martínez

AGRADECIMIENTOS

A mis Padres y Hermanos
que siempre están
conmigo apoyándome.

A mi asesor Ing. Marcelo Pérez,
al Mtro. Joaquín Meza,
profesores y amigos,
que siempre me ayudaron.

A todos ustedes, les doy las gracias
por su valioso apoyo en la culminación
de este proyecto y en mi formación
profesional.

Alejandro Villagómez Díaz

Introducción

Actualmente con los adelantos tecnológicos y científicos en el mundo podemos hacer diversas actividades con gran facilidad y en un tiempo menor que lo que nuestros antepasados ni siquiera imaginaron. No sólo en tiempo reducido sino que ni siquiera consideraron su factibilidad.

Vivimos en la era de las computadoras que encierra repercusiones favorables para cualquier área del conocimiento y de la vida cotidiana. Al igual estamos siendo testigos del inmenso crecimiento en la investigación, desarrollo y el uso de técnicas de visualización y graficación por computadora.

El campo de la visualización es muy amplio; a través de métodos, técnicas, sistemas y software se pueden tratar innumerables temas tales como: visualización de la información, animación y simulación, programación y lenguajes visuales y "rendering", entre otros.

Lo que más nos interesa en el desarrollo de esta tesis es la visualización de los datos proporcionados por el Laboratorio de Ingeniería Ambiental del Centro Tecnológico Aragón (CTA) perteneciente a la Escuela Nacional de Estudios Profesionales Aragón (ENEP Aragón). Por lo cual, la tesis se lleva acabo en colaboración con el mismo y la organización gubernamental de Municipio de Ciudad Nezahualcóyotl de ODAPAS que tiene el objetivo de monitorear los parámetros de los niveles de contaminación del agua en dicha comunidad. Como se sabe el agua es un elemento sumamente importante para la sobre vivencia del hombre.

El desarrollo del proyecto de tesis toma como base la captura de ciertos parámetros de contaminación recopilándose la información de 5 pozos, 3 cárcamos y 1 tanque de regulación, proporcionándonos 11 parámetros de cada uno: 1. el Ph, 2. Conductividad, 3. Salinidad, 4. Nitratos, 5. Color, 6. Coliformes fecales, 7. Turbiedad, 8. Temperatura, 9. Cloro, 10. Nitrógeno amoniacal, 11. Nitritos. Cada uno de ellos es comparado contra los límites permisibles aprobados y publicados por la Norma Oficial Mexicana MON-127-SSA-1994 aprobada en el Diario Oficial el 18 de enero de 1996.

Los Pozos de bombeo son estructuras conformadas por bombas de extracción que obtienen el agua del subsuelo o de un tanque de almacenamiento que es destinada a un lugar predeterminado.

Los Cárcamos de bombeo son estructuras metálicas y de concreto en forma rectangular o circular que recolectan el agua sucia proveniente de las alcantarillas, las fabricas, de casas habitación, entre otras. Para evitar inundaciones; dependiendo de la zona urbana en la que se encuentre será el tamaño de construcción de éste. El cárcamo está provisto de bombas de extracción para dirigir el agua recolectada hacia la red de drenaje profundo, a una planta de tratamiento o en su defecto al canal residual más próximo.

Los Tanques de regulación son estructuras rectangulares de material metálico o de concreto armado, que podemos encontrar en una forma superficial o enterrada. Se utilizan como medios de almacenamiento para regular el volumen y distribución del agua que se envía a las zonas conurbanas.

Una vez proporcionados y recopilados los datos de estos lugares son procesados y convertidos en gráficas que ilustran el comportamiento de los parámetros en el año, con gráficas de manera temporal y fijas así como con imágenes fijas y temporales apoyadas en una animación tanto de los parámetros de contaminación como del recorrido que realiza el Laboratorio de Ingeniería Ambiental por el Municipio de Ciudad Nezahualcóyotl, las cuales posteriormente son proporcionadas a ciertos usuarios e incluso al público en general.

Como el Web es un medio gigantesco de información, nos pareció apropiado elaborar una página en Internet para un mejor manejo de los datos, logrando de esta manera que la información se tenga de forma accesible y al alcance, en cualquier momento que se requiera.

Por lo expuesto con anterioridad, consideramos importante que nuestro tema de tesis fuera "VISUALIZACIÓN GRAFICA DE PARÁMETROS DE CONTAMINACIÓN EN POZOS Y DEPÓSITOS DE AGUA EN CIUDAD NEZAHUALCOYOTL".

Nuestra principal motivación para este tema de tesis es que como se sabe el agua es un elemento sumamente importante para la sobrevivencia del hombre y si apoyados con la visualización logramos proporcionar soluciones más eficientes e interpretaciones más exactas habremos logrado un proyecto muy útil.

Dividimos este trabajo en cinco capítulos, en el cual el primero hacemos referencia al concepto de visualización y al programa Pov-Ray que es la base del desarrollo de las gráficas y la animación. El modelado del plano de Ciudad Nezahualcóyotl con lo que conlleva como centros recreativos, colonias, avenidas, manzanas, ubicación de los puntos de recopilación de información cómo pozos, cárcamos, tanques, etc. Lo destinamos al capítulo dos. Mientras que el tercero lo dedicamos a la obtención y procesamiento de los datos proporcionados por el Centro Tecnológico Aragón y ODAPAS; desde la entrada de la información, procesamiento hasta la salida visualizada en forma de gráficas. Por lo que respecta a la creación de las animaciones de las gráficas y recorridos, así como los programas que se usan para la generación de las diversas secuencias se estableció en el capítulo cuatro y por último el capítulo cinco que contiene la elaboración, desarrollo y creación de una página Web para Internet.

Justificación

La elaboración del presente proyecto se hace con la finalidad de obtener el título de "Ingeniero en Computación" con un trabajo que muestre cómo la visualización por medio de imágenes y animaciones puede ser un medio de información mucho mas práctico en el razonamiento que una cantidad de textos escritos que vemos como información. Es constando que una persona tiene mucha más facilidad de captar, de entender y de recordar una imagen que varias hojas. Involucrando el concepto de animación en todos los sentidos ya que está es una simulación de un suceso real en forma virtual.

El tema nos interesó ya que hoy en día podemos ver empresas que utilizan técnicas de visualización por medio de animaciones para vender, comprar, enseñar, mostrar entre otros; no sólo información sino hasta productos e ideas de cualquier tipo. Aunado a que el costo de inversión y de producción es bastante pequeño.

Un problema que se tuvo en la realización de este proyecto fueron los recursos de hardware ya que llevar acabo un proceso de animación requiere de una gran cantidad de velocidad en procesamiento, almacenamiento y memoria, pero se logro hacer todo utilizando una Pc con AMD-K6-400 Mhz y software libre.

Objetivos

- Obtener el título de Ingeniero en Computación
- Facilitar el análisis y la interpretación de los parámetros de contaminación al usuario, en forma clara y comprensible por medio de representaciones gráficas y secuencias animadas.
- Crear un almacenamiento de información que permita la manipulación de información gráfica de manera dinámica.
- Mostrar la ubicación de las instalaciones y el recorrido que se lleva a cabo para la recopilación de información a través de una simulación en tercera dimensión.
- Dar a conocer de manera clara y comprensible, a los usuarios sobre la información obtenida en el transcurso de un año sobre los parámetros de contaminación que existen en el Municipio de Ciudad Nezahualcóyotl mediante una página Web.
- Reducir tiempos de espera en la toma de decisiones que realiza ODAPAS y el Centro Tecnológico Aragón.

I N D I C E

I. Visualización con Pov Ray

II. Modelado del Plano de Nezahualcóyotl

CAPÍTULO 1

Visualización con POV-RAY

1.1. Qué es Visualización

Ciertamente la visualización es aquella que se basa en técnicas para resaltar la información mediante representaciones gráficas, imágenes y animación, a través de las cuales podemos interpretar y representar datos para los usuario en forma clara y comprensible. Cantidades enormes de datos son convertidos en gráficas e imágenes. Los despliegues gráficos nos ayudan de manera considerable a entender la información compleja con mayor rapidez y eficacia que una descripción oral, escrita o tabulada, porque una impresión visual se comprende con mayor facilidad y requiere menos esfuerzo mental para averiguar acerca de los hechos. Esto reafirma el dicho de que "una imagen dice más que mil palabras".

La visualización permite utilizar las capacidades de análisis visual en el humano y así detectar los diferentes patrones de una manera mucho más efectiva.

Por lo anterior la visualización nos ayuda a incorporar los datos obtenidos de las muestras realizadas en los pozos y depósitos de manera visual ya que con lo que se cuenta actualmente es con la mera información, hojas y hojas de datos que se tiene que no nos refleja nada simples datos estadísticos, sin embargo con apoyo de lenguajes gráficos y lenguajes de programación podemos obtener gráficas y dibujos representativos de dicha información generando como respuesta un mayor razonamiento y entendimiento de lo que

ocurre en cada uno de los pozos y depósitos de Ciudad Nezahualcóyotl para tomar acciones de manera mas rápida y correcta.

1.2. ¿Qué es el trazado de rayos?

El trazado de rayos es una de las técnicas de generación de imágenes que permite la aplicación de un modelo de iluminación global para el cálculo de la iluminación en una escena.

El proceso que sigue el trazado de rayos es el siguiente. Se coloca un observador virtual dentro del espacio donde se define la escena, asociándole una dirección hacia donde va a ver y una amplitud de su campo visual. Una vez hecho esto, se puede establecer el conjunto de direcciones desde las cuales llega la información luminosa que compondrá la imagen. Este conjunto de direcciones es infinito por lo que hay que seleccionar un subconjunto finito que a fin de cuentas es lo que determinará la resolución de la imagen que generemos. A cada una de las direcciones seleccionadas les asociamos un rayo para el cual debemos calcular la iluminación que viene desde esa dirección. Si la dirección está dirigida a un objeto que emite luz, entonces la iluminación estará directamente relacionada con la emisión del objeto, pero si al seguir el rayo de luz llegamos a un objeto que no emite luz por si mismo, entonces debemos considerar otro tipo de fenómenos que ocurren en el punto donde se intercepta el rayo y la superficie del objeto. En este punto es necesario considerar las leyes físicas que rigen la interacción entre la luz y los materiales que componen los objetos, ya que dependiendo de las características del material y de las condiciones de iluminación del ambiente que rodea al objeto, será la luz que llegue hasta el observador en esa dirección.

Este es un proceso que puede ser extremadamente complicado pero que también nos permite generar imágenes con un alto grado de realidad puesto que se están empleado las leyes físicas que reflejan la realidad de lo que percibimos.

Por lo tanto en cada píxel en la imagen final se disparan uno o más rayos en la escena y se comprueba la intersección con cada uno de los objetos de la escena. Los rayos se originan en el observador, representado por la cámara, y pasan a través de la ventana de visión (que representa la imagen final).

Cada vez que el rayo golpea a un objeto, se calcula el color de la superficie en ese punto. Para ello se determina la cantidad de luz que proviene de cada fuente luminosa para ver si el objeto permanece en la sombra o no. Si la superficie es reflectora o traslúcida se trazan nuevos rayos para determinar la contribución de la luz reflejada o refractada en el color final de la superficie.

La figura muestra el proceso de visualización utilizando geometría en una escena simple de trazado de rayos.

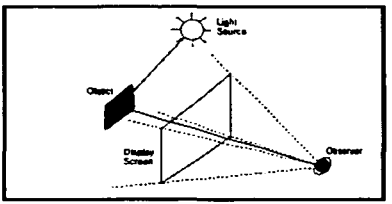


Fig. 1.1 Escena geométrica simple

En la siguiente figura se muestra una escena más compuesta con algunos objetos, fuentes de luz, sombras y algunos reflejos que han sido agregados.

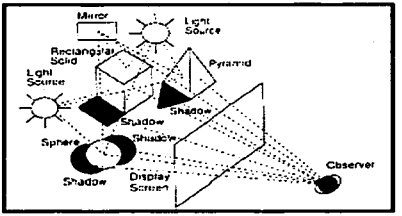


Fig. 1.2 Escena geométrica compuesta

Esto implica que la escena de tipo realista debe contener cientos de objetos y que el procesar todas las interacciones de rayos de luz para cada píxel sea una tarea laboriosa. Afortunadamente, solamente tenemos la posición de los objetos primitivos que definen una escena; las interacciones complicadas de luz y sombras son procesadas por el programa y esto hace posible trabajar en una computadora personal (PC).

A continuación se muestran algunos ejemplos de la aplicación del trazado de rayos en unos modelos sólidos con iluminación y uno con refracción:

Fig. 1.3 Modelo Sólido e Iluminación en una esfera

Fig. 1.4 Modelo Sólido e Iluminación en un cilindro

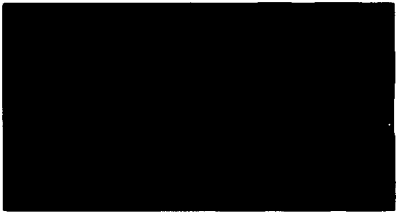


Fig. 1.5 Objetos en refracción

Algunas escenas más complicadas se muestran a continuación con un mayor detalle y de mejor calidad.

Fig. 1.6 Escenas de imágenes compuestas y de mayor calidad.[2.1]

1.3. Pov-Ray

Pov-Ray es una herramienta de representación en tercera dimensión (3D render) usada para crear sorprendentes imágenes foto realistas mediante escenas en 3D y en la creación de animaciones.

Para crear una imagen en POV-Ray, se necesita escribir un archivo de texto que codifique los objetos en la escena, usando un lenguaje especial de programación. El archivo de texto describe todos los aspectos que desearías tener en tu imagen final, tales como los de objetos simples como son: cilindros, esferas y otros objetos compuestos tales como montañas, planos, fractales, así como fuentes de luz, texturas, refracciones, puntos de vista (cámaras), etc.

La estructura de construcción del archivo de texto es semejante a la que se utiliza en el lenguaje de programación "C". Pov-Ray leerá este archivo de escena, y se generará la imagen basada en el código descrito.

Es necesario tomar en cuenta ciertas restricciones en el momento de escribir nuestra escena como son:

[2.1] Imágenes obtenidas de la página de Internet: http://www.gsl.net/on6jc/pov/

- Diferenciar entre letras mayúsculas y minúsculas.
- Definir explícitamente los procesos cuando inician y terminan con llaves, { }.
- No utilizar palabras reservadas del propio lenguaje.

El programa hace el primer proceso de análisis lexicográfico del o de los archivos que se tengan involucrados en la escena para determinar si están definidos correctamente los objetos y así generar el proceso de la reproducción de la imagen.

El tiempo de generación para recrear la escena en una imagen variará dependiendo de lo complicado que exista en la escena calculando el color de cada píxel en la imagen resultante, así como con los recursos de equipo de cómputo y sistemas operativos tales como: Ms-Dos, Windows, Linux, Unix, SunOS, Apple macintosh cada uno con sus requerimientos específicos. Si se desea obtener este software es gratuito y se puede obtener por vía Internet, por esto se llevo a cabo este proyecto con el fin de demostrar que siendo un software de fácil acceso, se pueden hacer cosas de muy buena calidad auxiliándose con otros programas que también son de distribución gratuita que describiremos en capítulos siguientes.

Pov-Ray es un programa escrito en lenguaje "C" estándar el cual puede ser compilado para correr sobre cualquier plataforma de cómputo.

Así mismo este programa puede incluir archivos tipo bibliotecas con extensión "inc" que se utilizan en lenguajes de programación a través de la directiva "include" que se han escrito previamente en el mismo lenguaje de Pov-Ray, con la cual nos permite crear nuevas librerías para los objetos, texturas y colores ya predefinidos. La forma de incluir estos archivos en nuestra escena es la siguiente:

```
#include "nombre_archivo.inc"
```

Ejemplo:

```
#include  "colors.inc"
#include  "textures.inc"
#include  "stones.inc"
```

1.3.1. Palabras Reservadas

Es importante conocer las palabras reservadas que utiliza el lenguaje Pov-Ray como cualquier otro, para no tener complicaciones en el momento de escribir el archivo de texto que contendrá nuestra escena. El programa brinda la opción de definir nuestras propias palabras reservadas para una mayor comodidad de la misma programación.

A continuación se listan las palabras reservadas:

adaptive	cone	image_map
agate	crand	include
agate_turb	cubic	interpolate
all	cylinder	intersection
alpha	declare	inverse
ambient	default	ior
area_hight	dents	jitter
background	difference	lambda
bicubic_patch	diffuse	leopard
blob	direction	level
blue	disc	light_source
bounded_by	distance	location
box	dump	look_at
bozo	fahloff	looks_like
brilliance	filter	mandel
bump_map	finish	map_type
bump_size	flatness	marble
bumps	fog	material_map
camera	frequency	max_intersections
checker	gif	max_trace_heveh
clipped_by	gradient	merge
clock	granite	metaliic
color	green	no_shadow
color_map	height_tield	normal
colour	hexagon	object
colour_map	iff	octaves
component		omega
composite		once

onion	rgb	tiles
open	rgbf	torus
phase	right	translate
phong	ripples	triangle
phong_size	rotate	turbulence
pigment	roughnes	type u_steps
plane	scale	union
point_at	sky	up
poly	smooth	use_color
pot	smooth_triangle	use_colour
quadric	specular	use_index
quartic	sphere	v_steps
quick_colo	spothight	version
quick_colour	spotted	water
radial	sturm	waves
radius	texture	wood
raw	tga	wrinkles
red	threshold	x
reflection	tightness	y
refraction	tile2	z

1.3.2. Características del Programa POV-Ray

Pov-Ray se caracteriza por lo siguiente:

- Lenguaje de descripción de escena fácil de usar.
- Archivos de imagen de muy alta calidad (hasta 48 bits de color).
- Visualización en IBM-PCs de 15 y 24 bits, usando el hardware apropiado.
- Fuentes luminosas dirigidas.
- Varios formatos de imagen, incluyendo Targa, PNG, PPM, BMP.
- Figuras primitivas básicas: esferas, cajas, cilindros, conos, triángulos y planos.
- Figuras primitivas avanzadas: toros, campos elevados, burbujas, triángulos suavizados, texto, fractales, etc.
- Geometría Constructiva Sólida (CSG).

A los objetos se les puede asignar materiales llamados texturas las cuales describen propiedades de la superficie como color, brillos, acabados, etc.

1.3.3. El Sistema de Coordenadas de POV-Ray

El sistema de coordenadas que utiliza POV-Ray es de tres dimensiones denotados por tres vectores perpendiculares entre si que se refieren a X, Y, Z. Será necesario para especificar un punto que se involucren a estos tres vectores de tipo real y que deben de estar denotados dentro de un picoparéntesis <>.

La forma de leer dicho sistema coordenado es tomar el sentido positivo del eje Y apuntando hacia arriba, el sentido positivo del eje X hacia la derecha y el sentido positivo del eje Z apuntando hacia el interior del monitor. Como se muestra a continuación:

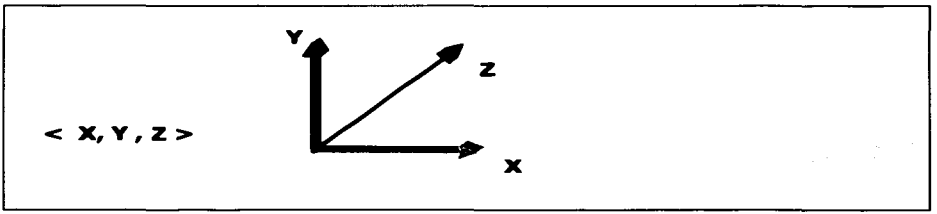


Fig. 1.7 Sistema Coordenado en el programa Pov-Ray

A este tipo de sistema de coordenadas se le conoce como el de "La Mano Izquierda", ya que al usar esta los dedos de la mano nos dan fácilmente la ubicación por que el pulgar apunta hacia la dirección positiva del eje X, el dedo índice hacia la dirección positiva del eje Y, y el dedo medio hacia la dirección positiva del eje Z.

1.3.4. El Formato de las Escenas en Pov-Ray

Como ya hemos visto una imagen foto realista en 3D se genera en Pov-Ray a través de un archivo de texto que contiene escrita una escena formada por varios objetos de figuras simples y compuestas, luces, cámaras, texturas, colores, etc., utilizando un método que se llama trazado de rayos.

Pov-Ray consta de un gran número de objetos y utilidades que juntos crean imágenes de calidad tipo realistas, pero por lo mismo solo nos enfocaremos a las utilizadas en el proyecto de la medición de paramentos de contaminación en el Municipio de Cd. Nezahualcóyotl. Si se necesita más información sobre todos los elementos que conforman el lenguaje de Pov-Ray se puede consultar:

Página de Internet: (http://www.arrakis.es/~pcostas/povray/)
Manual de Pov-Ray
Cd-Interactivo

1.3.4.1. Luces

Este elemento es uno de mucha importancia ya que es el objeto encargado de brindar la fuente de iluminación a nuestras escenas, sin esta los objetos no se verían, dando una imagen negra. Las fuentes de luz son invisibles y de dimensiones muy pequeñas, que al situarlas en un punto de la escena emite rayos luminosos que hacen visible a nuestros objetos.

Es muy importante saber colocar las fuentes de luz, por que estás crean brillos, reflejos y sombras sobre los otros objetos pudiendo ocasionar perdida de calidad y visibilidad de nuestra escena. En una misma escena podemos utilizar múltiples fuentes de luz, pero hay

que tomar en cuenta que esto generará un mayor tiempo en la renderización de la imagen.
La intensidad luminosa no disminuye con la distancia, lo cual se vera afectado un poco en el realismo de nuestra imagen, puesto que se sabe que la iluminación es inversamente proporcional al cuadrado de la distancia, pero que el efecto de los objetos luminosos sobre los cuerpos es debido al ángulo de incidencia, motivo por lo cual los rayos luminosos tienden a ser paralelos conforme aumente la distancia.

Para escribir una fuente de luz, su sintaxis es la siguiente:

```
light_source { < x, y , z> color nombre_color }
```

Como habíamos mencionado anteriormente Pov-Ray trabaja en sistema coordenado de tres dimensiones X, Y, Z los cuales tenemos que indicar en picoparéntesis <> para su colocación en la escena, por lo tanto la fuente de luz lleva sus tres valores de coordenadas más nombre_color que debe ser un color previamente definido en inglés, o bien si es necesario que el usuario defina un color se puede llevar a cabo por medio del indicador "rgb" y sus componentes de dicho color.

Por ejemplo; si necesitáramos una fuente de luz color azul en las coordenadas 10 en X, 5 en Y y 8 en Z, lo indicaremos de la siguiente forma:

```
light_source { < 10, 5 , 8> color rgb <0, 0, 1> }
```

Si utilizamos un color preestablecido en las mismas coordenadas será:

```
light_source { < 10, 5 , 8> color yellow }
```

1.3.4.2. Cámara

La escena después de ser iluminada, es necesario colocar un nuevo objeto llamado cámara que tiene por objetivo observar desde un punto dado todo lo que esta puesto en escena, la colocación de la cámara es de igual manera con tres coordenadas y entre picoparéntesis.

El punto de ubicación de la cámara es muy importante, ya que es el lugar en el espacio donde se comenzarán a observar y generar las imágenes en las cuales podremos ver todos los factores de la escena y si fuera necesario cuales no se enfocan de la misma.

La construcción de la cámara esta formado por dos vectores, el primero location <x, y , z> que ubica el lugar en el espacio desde donde se realiza la observación y el otro vector look_at <x, y, z,> que nos indica el punto a mirar de la escena. Los dos vectores son básicos y obligatorios dentro de la construcción de cualquier escena.

Su sintaxis es la siguiente:

```
camera {
        location < 3, 5, -10>
        look_at  < 0, 0,   0 >
        }
```

Opcionalmente se puede agregar dentro de la construcción de la cámara un elemento que se llama sky que producirá un efecto de cielo para una mejor vista al observador.

```
camera {
        location < 3, 5, -10>
        sky        < 1, 1    0>
        look_at  < 0, 0,   0 > }
```

1.3.5. Formas Simples

Para conocer mejor los objetos simples que utiliza Pov-Ray como esferas, conos, cilindros, cajas, se describe y construyen escenas individuales de cada una de las figuras para así mostrar una imagen de lo real que llega hacer, al mismo tiempo de lo fácil que es construir uno de estos elementos antes citados.

1.3.5.1. Esferas

La esfera es una figura simple siendo esta la más sencilla, su sintaxis es muy fácil de entender:

```
sphere  { < Centro >,  Radio }
sphere  { < x, y , z >,   R    }
```

En esta sintaxis la palabra <Centro> hace referencia al vector donde estará colocada la coordenada del centro de la esfera y R es el valor de tipo real que define al radio de la esfera.
Ejemplo:

```
sphere { <0, 0, 0>, .1
        pigment { Gold }
        finish {  ambient 0.1
                  diffuse 0.6
                  phong .75
                }
       }
```

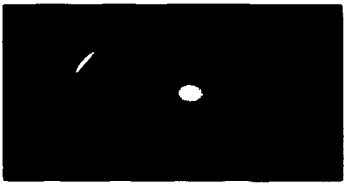


Fig. 1.8 Esfera en Pov-Ray

Aquí nos muestra la esfera con centro en el origen y 1 unidad de radio, hay que hacer notar que la coordenada del centro esta separada con una coma (,), además hay que agregar la palabra reservada "pigment" para asociarle un color al objeto que en este caso es el oro, y así poder observar la esfera con dicho color ya sea en pantalla ó como escena terminada.

1.3.5.2. Cajas

Es otro objeto de mucha utilidad en la construcción de escenas, la forma de trabajar con estos objetos es la siguiente:

```
box { <Esquina_inf>, <Esquina_sup> }
box {   < x, y, z >  ,  < x1, y1, z1 > }
```

Donde Esquina_inf y Esquina_sup son vectores de posición que definen esquinas opuestas, el primer vector debe tener valores menores que de los del segundo vector, debido a que las cajas solo pueden ser definidas con sus aristas paralelas a los ejes coordenados.

Ejemplo:

```
box { <-1, -1, -1>, <2, 0, 1>
        pigment { MediumAquamarine }
        finish  { ambient 0.1
                  diffuse 0.6
                  phong .0175
                }
    }
```

Fig. 1.9 Caja en Pov-Ray

1.3.5.3. Cono

Este objeto lo construimos tomando en consideración las coordenadas del centro de la base y el centro de coordenadas de la otra base que en este caso sería la altura, a ambos vectores se les agrega un radio para darle la forma que se desea.

Ejemplo:

```
cone { <0, 1, 0>, 0.3
       <1, 1, 4>, 1.0
      open
       texture{Sandalwood scale .3
                  finish {  ambient 0.1
                            diffuse 0.6
                            phong .5175
                         }
                }
       }
```

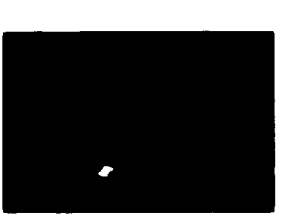


FIg 1.10 Cono en Pov-Ray

A este ejemplo se agrego la palabra "open" que es utilizada para no dibujar las bases, como es mostrado en la figura.

1.3.5.4. Cilindro

La sintaxis de esta objeto es:

cylinder { <Centro_1>, <Centro_2>, Radio }

Los vectores "Centro_ 1" , "Centro_2" son las coordenadas de los centros de cada base del objeto y R el radio de la figura.

Ejemplo:

```
cylinder { <1, -6, 1>,<1, 5, 1>, 1.25
           open
           pigment { YellowGreen }
          finish     {  ambient 0.1
                        diffuse 0.6
                        phong .575
                      }
          }
```

Fig. 1.11 Cilindro en Pov-Ray

1.3.5.5. Plano

Estos objetos son de extensión infinita que utiliza Pov-Ray en forma muy sencilla manejándolo en forma de un vector. Los componentes que lo forman son:

```
Plane { <vector> Factor
        Pigment { Color }
      }
```

Pov-Ray tiene predefinidos ya los vectores para tener un mejor manejo de los planos y definiéndolos con los indicadores <1,0,0> para X, <0,1,0> para Y, <0,0,1> para Z, esto se ejemplifica como Plane { y, 0 Pigment { Checker color Red, color Blue } }

También un plano lo podemos representar a través de una expresión polinomial que se denota de la siguiente forma: Ax + By + Cz + D =0 y la expresamos como un objeto de Pov-Ray utilizando la siguiente sintaxis: plane { < A, B, C >, D }

1.3.6. Texturas y Pigmentos

Después de crear la escena compuesta con cámaras, fuentes de luz, objetos simples, será necesario para tener una semejanza más real con la vida cotidiana describir la apariencia de la superficie, es decir darle el acabado foto realista a nuestras figuras como color, brillo, rugosidad, sombras y propiedades específicas de un objeto en particular.

Pov-Ray para poder dar este acabado a lo antes citado utiliza texturas predefinidas en unos archivos include estándar que son librerías que contienen valores definidos para cada una de las texturas, estos archivos son "texturas.inc" y "stones.inc"

La palabra texture indica que al objeto se le adicionara un acabado por medio de una textura ya sea previamente definida ó por una definición del usuario a través de las siguientes opciones.

```
texture {
        Tipo_Textura
        pigment {   }
        normal  {   }
        finish  {   }
       }
```

Pigmento. Es una sentencia donde están definidos todos los patrones y colores estándares que se aplican a una textura.

El tipo más simple de pigmento que se utiliza es color sólido, que con solo escribir el color específico puede aplicarse a un objeto, para esta gama de colores se utiliza un archivo llamado "colors.inc":

```
pigment  { color Pink}
```

Otra manera práctica de aplicar una textura con un color sólido no especificado en el archivo colors.inc de Pov-Ray es utilizar el modo rgb el cual nos proporciona una enorme gama de posibilidades y combinaciones de colores a utilizar sin preocuparnos por este tema.

```
pigment  rgb <.25, .25, .25>
```

A pesar de la dimensión de colores a la que tenemos acceso a crear solo es posible grabar 16.7 millones de colores.

Para tener un acabado casi real como abolladuras, óxidos, brillos, entre otros. Se utiliza dentro de las opciones del bloque texture con parámetro "finish" como lo observaremos a continuación con tres esferas.

```
sphere{<-2.8, 0, 0>, 1
      texture{
         pigment {White_Wood}
         scale 1.5
         finish {Glossy}
       }
    }

sphere{<0, 0, 0>, 1
      texture{  Brushed_Aluminum
         scale .5
         finish {Dull}
       }
    }

sphere{<2.8, 0, 0>, 1
      texture{
        pigment {Jade}
        finish {Phong_Glossy}
       }
    }
```

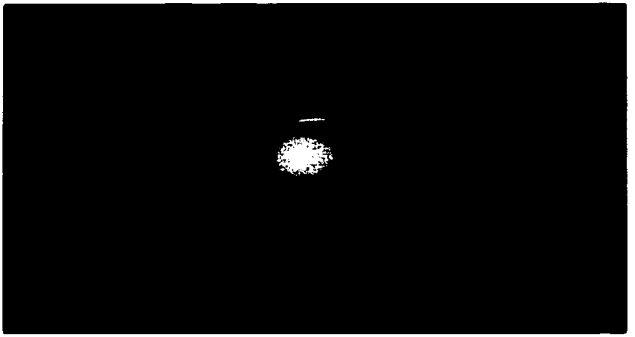


Fig. 1.12 Diferentes tipos de texturas sobre 3 esferas

1.3.7. Parámetros de Renderización

La generación de imágenes de calidad fotográfica de escenas producidas en Pov-Ray utiliza algunos de estos parámetros que se describen a continuación:

Se obtiene un archivo BMP de acuerdo al formato de salida (Ancho x Largo medido en píxeles)

La tabla siguiente muestra algunos valores de renderización:

+Annn	Produce una imagen con suavizados (Anti-alias de 0.0 a 0.3), entre más pequeños sean los valores más suavizado se tendrá.
+A	Tiene un valor por default de 0.3 de suavizado
-A	Apaga el anti-alias
+C	Termina una imagen que este abortada
+Dxxx	Muestra la generación de la imagen mientras se procesa
+Hnnn	Altura en píxeles de la imagen
+Wnnn	Ancho en píxeles de la imagen
+Qn	Diferentes calidades de la imagen de 0 a 9
+FT	Formato de salida TGA
+FD	Formato Dump
-F	Deshabilita la opción de salida a un archivo

La siguiente escena mostrada contiene varios objetos que hasta el momento hemos analizados.

Ejemplo.
La siguiente imagen concentra varias figuras simples en una sola imagen más compleja, incluyendo luces, cámara, texturas, colores, etc.[1.2]

Fig. 1.13 Grados de complejidad realizada en Pov-Ray

1.3.8. Transformaciones

Al tener ya figuras con objetos simples definidos, podemos moldear a estos objetos para que se acoplen a una situación más descriptiva, ya que no todo lo que observamos en nuestra realidad es de una forma cuadrada, por esto mismo se les puede aplicar una serie de transformaciones como trasladarlos de lugar, escalarlos o rotarlos, y al mismo tiempo un objeto lo podemos solo trasladar o bien escalar, rotar o cualquier otra combinación que se adapte a nuestras necesidades.

[1.2] El código para generar esta imagen se encuentra en el Manual de Pov-Ray. Persistence of Vision, Autor Pablo Costas. Diciembre 1997. España

1.3.8.1. Traslación

A cualquier objeto de los antes citados podemos trasladarlos de una posición a otra, la cual se indica por medio de un vector.

Su sintaxis es: translate <x, y, z>

Ejemplo:

```
sphere { <0, 0, 0>, 2.5
         texture {
                   pigment {NewTan}
                   finish {Phong_Glossy }
                  }
        }
```

```
sphere { <0, 0, 0>, 2.5
         texture {
                   pigment {NewTan}
                   finish {Phong_Glossy }
                  }
         translate<-8,-8,-20>
        }
```

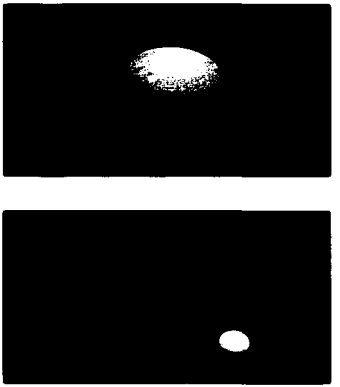


Fig. 1.14 Translación de una esfera

1.3.8.2. Escala

Esta opción se aplica a todos los objetos con el fin de cambiar su tamaño con respecto a cualquiera de sus ejes, ya sea en conjunto o independientemente de los valores del vector <x,y,z>. Para reducir se debe utilizar un factor menor a 1.0 y para aumentar un factor mayor a 1.0.

Su sintaxis es: scale <x, y, z>

1.3.8.3. Rotar

En todas las figuras que hemos mencionado podemos cambiar la dirección por medio de una rotación la cual se aplica a un eje o a todos a la vez, esta rotación se mide en grados.

Su sintaxis es: rotate <x, y, z>

Ejemplo: En este se ejemplifican las 2 últimas transformaciones.

```
cylinder { <0, -3, 0>, <0, 4, 0> 2.5
           open
           texture { Tan_Wood finish {Phong_Glossy } }
           rotate <45, 0, -45>
          }
```

```
cylinder { <0, -3, 0>, <0, 4, 0> 2.5
           open
           texture { Tan_Wood finish {Phong_Glossy } }
           scale <0.25, 0, 0.5>
           rotate <45, 0, -45>
          }
```

Fig. 1.15 Rotación de un cilindro

1.4. Geometría Sólida Constructiva

La Geometría Sólida Constructiva (CSG) es una poderosa herramienta que se utiliza para combinar figuras primitivas simples para crear objetos más complejos.

En Pov-Ray la Geometría Sólida Constructiva (CSG) nos permitirá construir figuras mucho más complejas a partir de las figuras básicas que hemos estudiado en este capítulo, esto es combinando las formas primitivas a través de operaciones que se llevan entre ellas, lo que con lleva a generar una producción de imágenes muchos más apegadas a la realidad en la que nos encontramos, dichas operaciones son: la unión, dos o más formas se asocian para ser utilizadas como una sola pieza. Otra es la intersección en la que sólo aparece el volumen común a todas las figuras. Y por último la diferencia que es análoga a la intersección.

Los objetos CSG pueden ser extremadamente complejos, ya que pueden estar anidados profundamente, es decir una unión puede contener varias intersecciones, y éstas varias diferencias y así las combinaciones que nosotros necesitamos para nuestras escenas.

Para mayor conocimiento de estas operaciones consultar el libro: Álgebra para Preuniversitarios. Rodolfo Alvarado. Edit. Esfinge. Así como libros de Lógica.

Al aplicar las operaciones a varios objetos logramos obtener las siguientes ventajas:

- Las escenas son interpretadas más rápido, ya que el ordenador solo leerá un solo objeto a la vez.
- Se obtiene un ahorro en memoria debido a que el programa construye un solo modelo de textura para un solo objeto, mientras que si se la aplicamos a objetos simples individuales a pesar de tener características iguales se realiza una copia independientemente para cada uno ellos.
- El cambiar las propiedades en la decisión de la apariencia del objeto es más fácil a un solo objeto compuesto por CSG que a muchos individuales.

1.4.1. Operaciones en Pov-Ray

1.4.1.1 Unión

Esta es la primera operación entre objetos que utiliza Pov-Ray mediante la Geometría Sólida Constructiva (CSG) que permite generar imágenes complejas a partir de 2 o más objetos simples, uniéndolos entre si para formar un sola figura objeto con propiedades específicas únicas para toda ella.

```
union{
      sphere   {<0, 0, 0>, 1
                scale<1.2, 0.5, 0>}
      cylinder {<0, -2, 0> <0, 2, 0>, 0.7 }

    //utilidad de la Unión

    texture{White_Wood}
    rotate<35, 0, 50>

 }
```

Fig. 1.16 Unión entre una esfera y un cilindro

1.4.1.2. Intersección

La operación de la intersección es similar a la función lógica que se utiliza en matemáticas de disyunción "^", lográndose como resultado una figura nueva que es el producto de la intersección de los puntos que pertenecen a dos o más objetos, haciendo desaparecer lo que no pertenezca a esta operación.

```
intersection{
          sphere   {<0, 0, 0>, 1
                    scale<1.2, 0.5, 0> }
          cylinder {<0, -2, 0> <0, 2, 0>, 0.7 }

  //utilidad de la Intersección

  texture{White_Wood}
  rotate<35, 0, 50>

}
```

Fig. 1.17 Intersección entre una esfera y un cilindro

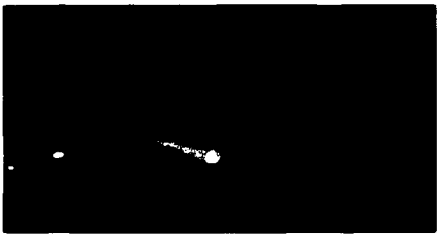
1.4.1.3. Diferencia

La figura compleja resultante de esta operación al aplicarse a dos objetos son todos los puntos de coordenadas que pertenecen a la primera figura pero no a la segunda.

```
difference{
        sphere  {<0, 0, 0>, 1
                 scale<1.2, 0.5, 0>}
        cylinder {<0, -2, 0> <0, 2, 0>, 0.7 }

  //utilidad de la Diferencia

  texture{White_Wood}
  rotate<35, 0, 50>

 }
```

Fig. 1.18 Diferencia entre una esfera y un cilindro

CAPÍTULO 2
Modelado del Plano de Nezahualcóyotl



2.1. Obtención del Plano del Municipio de Cd. Nezahualcóyotl

Para realizar el modelo del plano de Cd. Nezahualcóyotl y el Municipio de Aragón en tercera dimensión utilizamos su área de influencia y alguna simbología que viene marcada en la Guía Roji® del Distrito Federal abarcando el Municipio de Nezahualcóyotl.

Para esto se bajó del portal de Internet de la página del Guía Roji® los segmentos de cada colonia. Se agruparon y se formo un plano a una escala considerable que se tomó como base para las medidas y detalles para lograr el modelo en 3D del plano.

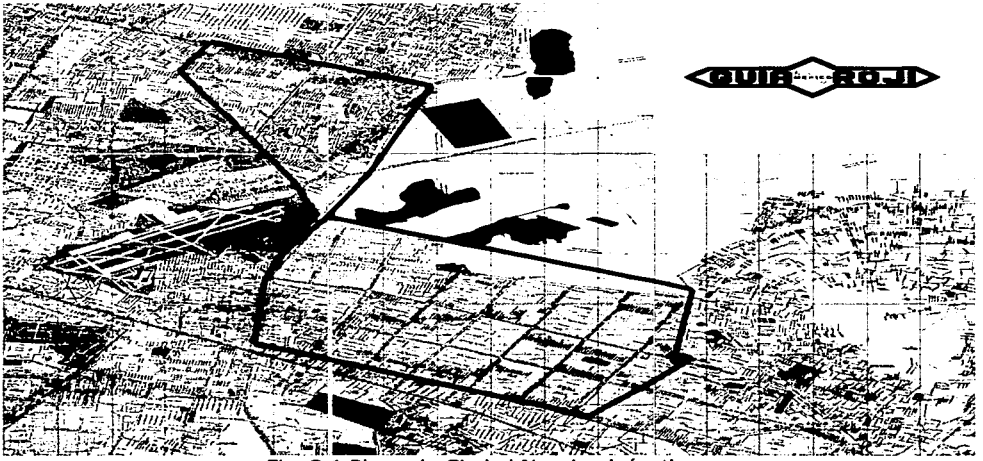


Fig. 2.1 Plano de Ciudad Nezahualcóyotl

Esta similitud se llevó a cabo a través de rectángulos que simulan manzanas que fueron formándose en conjunto para dar colonias para así poder tener una ubicación real de lo que está marcado en la Guía Roji®, al mismo tiempo se utilizó la misma figura para hacer el trazo de calles y avenidas principales para así tener una similitud lo más real posible.

Se utilizó la simbología marcada en la Guía Roji® como los hospitales, áreas verdes, supermercados, iglesias, escuelas y del estadio agregando los pozos, cárcamos y tanque de regulación en donde es recopilada la información para el análisis en dicho proyecto.

Estas figuras antes citadas están representadas en tercera dimensión utilizando Pov-Ray en una forma muy sencilla y rápida, principalmente las que se refieren a la simbología, mientras que en el modelado de los objetos a los que se hace referencia para el análisis y recopilación de información, nos proporcionaron unas fotos de las cuales se procedió a realizar su modelo en 3D utilizando Pov-Ray y CGS que se describen más adelante.

Para dar por terminado el modelo del plano tridimensional se conformaron las manzanas, avenidas y calles colocando sobre estás los modelos de la simbología y puntos de análisis.

En este plano por las características de estar en 3D nos brinda la facilidad de poder hacer una animación en la cual trazamos y mostramos el recorrido que se llevó a cabo por el Laboratorio de Ingeniería Ambiental, perteneciente al CTA de la ENEP Aragón para obtener la información referente a los parámetros de contaminación del agua. Este recorrido se ejemplificó utilizando un cochecito que va entre las avenidas (Capitulo IV).

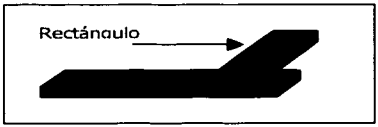
2.2. Modelado en 3D del Plano

Para llevar a cado el desarrollo del plano en 3D de Cd. Nezahualcóyotl, se utilizo como figura principal, una caja en forma de rectángulo, asignándole el color negro que muestra las avenidas principales, mientras que otro rectángulo, de color azul representa las manzanas que conforman el municipio.

Para darle forma al diseño del modelo tridimensional se fue construyendo a través de varias uniones de cajas, tanto para las avenidas como para las manzanas, acoplándose a éstas conos y esferas, para así dar una mejor descripción del trazo del Municipio.

Este modelo sólo abarcará la ubicación del Campus Aragón, sus manzanas aledañas y lo concerniente al municipio de Cd. Nezahualcóyotl que es principalmente el área del proyecto. Se muestra a continuación la construcción de manzanas y avenidas para formar el Municipio de Nezahualcóyotl y la parte alta de Aragón.

2.2.1. Avenidas



La construcción de las avenidas está realizada por la unión de varios rectángulos con propiedades de color negro, apoyándonos de figuras como conos y esferas para detallar mejor el trazo del plano. Esto nos va dando la forma del esqueleto por así llamarlo, en donde colocaremos las manzanas, la simbología y los puntos de información que son los pozos, tanques y cárcamos.

Las avenidas principales son:

Horizontales:
Av. Texcoco, Av. Pantitlán, Av. Chimalhuacán, Av. Bordo de Xochiaca.

Verticales:
Av. Periférico, Av. Riva Palacios, Av. Nezahualcóyotl, Av. Gral. Vicente Villada, Av. Carmelo Pérez, Av. López Mateos, límite del Municipio Canal la Campana.

Se muestra a continuación el plano con las avenidas trazadas desde la ENEP Campus Aragón indicado por el Escudo de la UNAM y las Torres representativas a dicha Escuela, hasta la última avenida del Municipio.

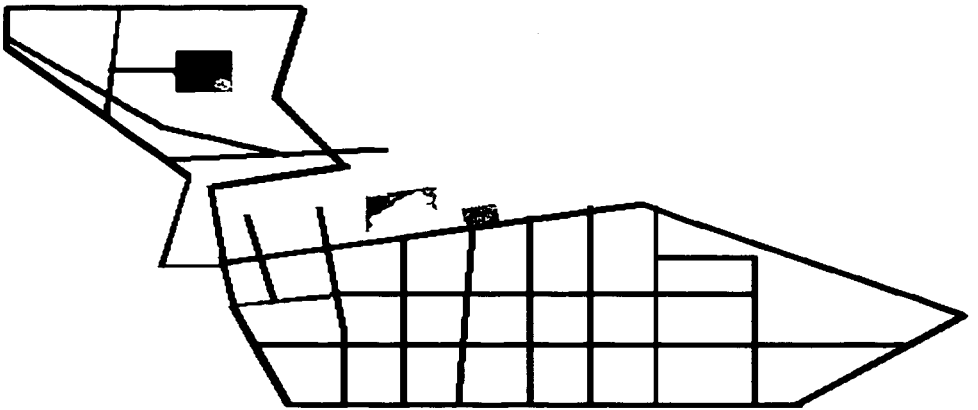


Fig. 2.2 Vista aérea del plano en 3D del proyecto.

2.2.2. Manzanas y Colonias

Para la representación de una sola manzana se utilizó un rectángulo de color azul, por lo tanto para la conformación de las colonias sólo se fueron repitiendo estos rectángulos tomando en cuenta la forma y el estilo que están marcados en la Guía Roji®. En ésta se contaron las manzanas que existen entre una avenida y otra, para así ir colocando los rectángulos azules de la misma manera, se utilizó uniones e intersecciones entre los mismos rectángulos, y algunas veces conos para asemejarse con lo marcado en la Guía Roji®.

El código del rectángulo es:

```
Box {<0,0,0,> ,<1,1,1>
    pigment { Blue } }
```

Para la facilidad de la colocación de las manzanas se utilizaron sentencias del tipo If, While y For que tienen la misma funcionalidad de cualquier Lenguaje de Programación, recordando que Pov-Ray es también un lenguaje de programación de imágenes foto realistas.

Fig. 2.3 Colocación de rectángulos para generación del plano

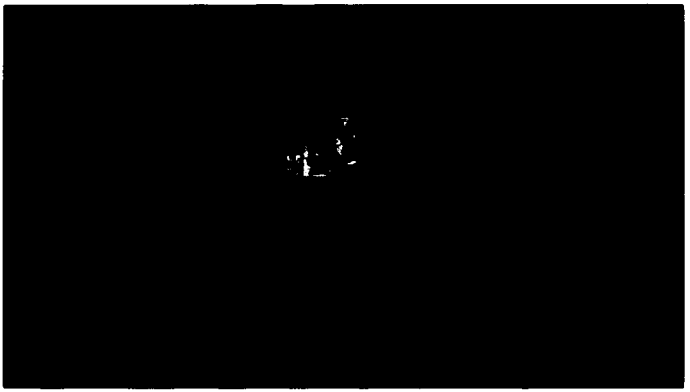


Fig. 2.4 Colocación de rectángulos para generación del plano

También por medio de rectángulos con propiedades color verde y otros con textura de tipo pasto, nos auxiliamos con esferas y conos para hacer diferencias e intersecciones, colocamos las áreas verdes que más sobresalieron como se muestra en la figura anterior.

El código de una área verde puede ser:

```
box {<0,0,0,> ,<1,.25,2>
        pigment { Green } }
sphere {<0,0,2>,.15
        pigment { Black} }
```

Después de analizar el trazo del Municipio se observó que muchas colonias y calles son semejantes en su diseño y construcción, facilitando el montaje de las manzanas a través de las sentencias antes citadas, ahorrándonos un considerable tiempo en el término del plano en 3D.

Fig. 2.5 Plano en 3D del Proyecto de Tesis, Cd. Nezahualcóyotl

Se consideró colocar el nombre de la mayoría de las colonias y avenidas principales con el objetivo de ubicarse en el momento de llevar a cabo las diferentes actividades del proyecto.

Fig. 2.6 Plano en 3D del Proyecto de Tesis, Parte de Aragón

2.3. Construcción de la Simbología

Los modelos mostrados en este proyecto son construidos utilizando el lenguaje de Pov-Ray antes citado, tomando como base la simbología marcada en la Guía Roji® del Distrito Federal.

Estos modelos en 3D como veremos a continuación se construyen de una manera muy fácil, rápida y sencilla utilizando sólo figuras básicas que en cierto momento se van complicando poco a poco dependiendo del modelo a diseñar. Esto lo mostraremos a continuación:

2.3.1. Iglesia

```
// ************************************

#include "colors.inc"

 camera{location<-1.5,4,15>
              look_at<-1.5,3.5,0> }

light_source{<-1.5,4,25>color rgb<1,1,1>}
light_source{<7,9,15>color rgb<1,0,1>}
light_source{<-7,9,15>color rgb<1,0,1>}
light_source{<-1.5,15,-1.5>color rgb<1,1,1>}

#declare cruz=union{   box{<0,0,0><-3,2,-3>}
                      box{<-1,2,-1><-2,7.5,-2>}
                      box{<0,5.5,-1><-3,6,-2>}  }
```

Fig. 2.7 Iglesia

2.3.2. Hospital

```
// ************************************

#include "colors.inc"

camera{location<0,14,22>
                  look_at<0,3.5,5>}

light_source{<0,15,25>color rgb<1,1,1>}
light_source{<7,5,15>color rgb<1,1,1>}
light_source{<-7,5,15>color rgb<1,1,1>}
light_source{<0,5.5,0>color rgb<1,1,1>}

#declare cruzr= union{box{<3,3,1><-3,-3,-1>}
                 union{box{<1.1,3.1,1.1><-1.1,-3.1,-1.1>}
                       box{<3.1,1.1,1.1><-3.1,-1.1,-1.1>}
                       pigment{Red}}
               pigment{White}
```

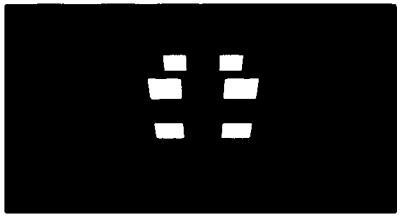


Fig. 2.8 Hospital

```
                scale<1,1,0.1>
               }
 #declare cruz_roja= union{object{cruzr}
          object{cruzr   rotate<0,90,0>   translate<3,0,-3>}
          object{cruzr   rotate<0,-90,0>  translate<-3,0,-3>}
          object{cruzr   translate<0,0,-6>}
          object{cruzr   rotate<90,0,0>   translate<0,-3,-3>}
         }
```

2.3.3. Virrete

```
// ***********************************

#include "colors.inc"

camera{location<0,4,18>
                look_at<0,2,5>   }

light_source{<0,1,20>color rgb<1,1,1>}
light_source{<8,-3,5>color rgb<0,1,1>}
light_source{<-8,-3,5>color rgb<0,1,1>}
light_source{<0,8,0>  color rgb<0,1,1>}

#declare virrete=union{box{<5,0.5,5><-5,0,-5>
                        rotate<0,45,0>}
                       difference{sphere{<0,-2.5,0>4.5}
                            box{<5,5,5><-5,0,-5>}
                            sphere{<0,-4.5,0>4.1
                              scale<1.5,1,1.2>}
                       }
                  pigment{Blue}
                  translate<0,2,0>
                 }
```

Fig. 2.9 Virrete

2.3.4. Áreas Verdes

```
// ************************************

#include "colors.inc"

camera{location<0,4,18>
                look_at<0,2,5>   }

light_source{<0,1,15>color rgb<1,1,1>}
light_source{<7,5,10>color rgb<.51,.3,.51>}
light_source{<-7,5,10>color rgb<.51,.33,.51>}

#declare muneco=union{sphere{<0,4.5,0>0.5}
                sphere{<0,5.5,0>0.8}
                cylinder{<0,0,0>,<0,4,0>,0.8}
                cylinder{<0,0,0>,<2,-4,0>,0.5}
                cylinder{<0,0,0>,<-2,-4,0>,0.5}
                cylinder{<-2,4,0>,<2,4,0>,0.4}
                cylinder{<-2,4,0>,<-2,6,0>,0.3}
                cylinder{<2,4,0>,<2,6,0>,0.3}
                sphere{<2,4,0>0.3}
                sphere{<-2,4,0>0.3}
                pigment{Green}
               }
```

Fig. 2.10 Áreas Verdes

2.3.5. Súper Mercado

```
// ***********************************

#include "colors.inc"
camera{location<0,10,30>     look_at<0,3.5,5> }

light_source{<0,10,30>color rgb<1,1,1>}
light_source{<9,15,10>color rgb<1,1,1>}
light_source{<-9,15,10>color rgb<1,1,1>}
light_source{<9,15,-10>color rgb<1,1,1>}
light_source{<0,11,-2>color rgb<1,1,1>}
light_source{<0,15,-2>color rgb<1,1,1>}


#declare pata=union{cylinder{<0,0,0>,<0,-4.5,0>,0.5     pigment{Red}}
                   cylinder{<0,-4.5,0.7>,<0,-4.5,-0.7>,0.8     pigment{Green}}
                   cylinder{<0,-4.5,0.75>,<0,-4.5,-0.75>,0.3   pigment{White}}}
#declare agujeros=union{box{<0.5,3.5,5><-0.5,4.5,-5>}
                        box{<0.7,3.5,5><1.7,4.5,-5>}
                        box{<1.9,3.5,5><2.9,4.5,-5>}
                        box{<3.1,3.5,5><4.1,4.5,-5>}
                        box{<4.3,3.5,5><5.3,4.5,-5>}
                        box{<-0.7,3.5,5><-1.7,4.5,-5>}
                        box{<-1.9,3.5,5><-2.9,4.5,-5>}
                        box{<-3.1,3.5,5><-4.1,4.5,-5>}
                        box{<-4.3,3.5,5><-5.3,4.5,-5>}    }

#declare agujeros1=union{box{<-7,3.5,0.5><7,4.5,-0.5>}
                         box{<-7,3.5,0.7><7,4.5,1.7>}
                         box{<-7,3.5,1.9><7,4.5,2.9>}
                         box{<-7,3.5,-0.7><7,4.5,-1.7>}
                         box{<-7,3.5,-1.9><7,4.5,-2.9>}    }

#declare carrito=union{difference{box{<-6,0,4><6,8,-4>}
                                  object{agujeros}
                                  object{agujeros    translate<0,1.2,0>}
                                  object{agujeros    translate<0,2.4,0>}
                                  object{agujeros    translate<0,-1.2,0>}

                                  object{agujeros    translate<0,-2.4,0>}
                                  object{agujeros1}
                                  object{agujeros1   translate<0,1.2,0>}
                                  object{agujeros1   translate<0,2.4,0>}
                                  object{agujeros1   translate<0,-1.2,0>}
                                  object{agujeros1   translate<0,-2.4,0>}
                                  box{<-4.5,0.5,3.5><4.5,11,-3.5>}        }
                        union{cylinder{<0,0,-4>,<0,0,4>,0.5}
                              cylinder{<0,0,-3.5>,<-2.5,-2.5,-3.54>,0.5}
                              cylinder{<0,0,3.5>,<-2.5,-2.5,3.5>,0.5}
                              translate<8,10,0>                                  }
                        object{pata   translate<4.5,0,3.5>}
                        object{pata   translate<4.5,0,-3.5>}
                        object{pata   translate<-4.5,0,3.5>}
                        object{pata   translate<-4.5,0,-3.5>}
                        cylinder{<-4.5,-2,3.5>,<0,0,3.5>,0.4}
                        cylinder{<0,0,3.5>,<4.5,-2,3.5>,0.4}
                        cylinder{<-4.5,-2,-3.5>,<0,0,-3.5>,0.4}
                        cylinder{<0,0,-3.5>,<4.5,-2,-3.5>,0.4}
                        pigment{Red}
                }
```

Fig. 2.11 Súper Mercado

2.3.6. Estadio

```
// ************************************
```

```
#include "colors.inc"

camera{location<0,15,25>
      look_at<0,3.5,5>}

light_source{<0,15,20>color rgb<1,1,1>}
light_source{<9,5,10>color rgb<1,1,1>}
light_source{<-9,5,10>color rgb<1,1,1>}
light_source{<0,20,0>color rgb<1,1,1>}


#declare lampara=union{cylinder{<0,0,0><0,5,0>,0.75
                        pigment{Gray}}
                  union{box{<2.5,5,0.5><-2.5,8,-0.5>
                        pigment{Gray}}
                      box{<2.25,5.25,0><-2.25,7.75,-0.6>
                        pigment{Yellow}}
                      rotate<-25,0,0>
                      translate<0,0,1.8>}
                   scale<0.4,0.4,0.4>
                  }


#declare cancha=union{box{<10,0.5,5><-10,0,-5>
                        pigment{Green}}                    //pasto
               difference{box{<10.2,0.5,5.2><-10.2,0,-5.2>}
                          box{<10,0.6,5><-10,0,-5>}
                          pigment{White}}                  //orillas
                     union{difference{cylinder{<0,0,0><0,0.6,0>,2}
                                      cylinder{<0,0.1,0><0,0.7,0>,1.8}} //circulo central
                           sphere{<0,0,0>,0.25
                                  scale<1,0.3,1>
                                  translate<0,0.6,0>}      //inicio
                           box{<0.1,0.6,5><-0.1,0,-5>}     //linea central
                           box{<6.6,0.6,2.5><6.4,0,-2.5>}
                           box{<6.5,0.6,2.6><9.99,0,2.4>}
                           box{<6.5,0.6,-2.6><9.99,0,-2.4>}
                           difference{cylinder{<0,0,0><0,0.6,0>,2}
                                      cylinder{<0,0.1,0><0,0.7,0>,1.8}
                                      box{<0,2,3><-3,-2,-3>}
                                      translate<-6.5,0,0>} //area grande izq
                           box{<8.6,0.6,1.5><8.4,0,-1.5>}
                           box{<8.5,0.6,1.6><9.99,0,1.4>}
                           box{<8.5,0.6,-1.6><9.99,0,-1.4>}
                           sphere{<0,0,0>,0.25
                                  scale<1,0.3,1>
                                  translate<-7.5,0.6,0>}   //area chica izq
                           box{<-6.6,0.6,2.5><-6.4,0,-2.5>}
                           box{<-6.5,0.6,2.6><-9.99,0,2.4>}
                           box{<-6.5,0.6,-2.6><-9.99,0,-2.4>}
                           difference{cylinder{<0,0,0><0,0.6,0>,2}
                                      cylinder{<0,0.1,0><0,0.7,0>,1.8}
                                      box{<0,2,3><3,-2,-3>}
                                      translate<6.5,0,0>}  //area grande der
                           box{<-8.6,0.6,1.5><-8.4,0,-1.5>}
                           box{<-8.5,0.6,1.6><-9.99,0,1.4>}
```

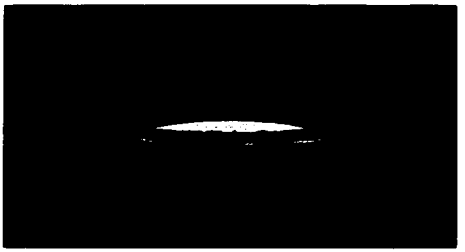


Fig. 2.12 Estadio de frente

Fig. 2.13 Perspectiva del Estadio

```
                box{<-8.5,0.6,-1.6><-9.99,0,-1.4>}
                sphere{<0,0,0>,0.25
                     scale<1,0.3,1>    translate<7.5,0.6,0>}  //area chica der
                     pigment{White}}  }

#declare estadio=union{object{cancha
                  scale<0.5,0.5,0.75>}
            difference{cone{<0,0,0>,3.8<0,3,0>,5}
                     cone{  <0,- 0.1,0>,3.1<0,3.5,0>,4.8}
                     pigment{Gray}
                     scale<2.4,2.1,1.75>}
            difference{sphere{<0,-0.1,0>,3.1}
                     object{cancha}
                     difference{cone{<0,0,0>,3.3<0,3,0>,5}
          cone{<0,-0.1,0>,3.1<0,3.5,0>,4.8}}
                     scale<2.4,0.03,2.1>
                     pigment{Green}}
            text{ttf "timrom.ttf","NEZA 86",2,0
                pigment{Green}
                scale<1.5,1.5,0.7>
                rotate<0,180,0>
                translate<3.35,6,9.4>}
            object{lampara     translate<5,6,7.5>}
            object{lampara     translate<-5,6,7.5>}
            object{lampara     rotate<0,180,0>   translate<5,6,-7>}
            object{lampara     rotate<0,180,0>  translate<-5,6,-7>}
            difference{cone{
                     <0,-0.1,0>,3.1<0,3.5,0>,4.8}
                     cone{<0,-2,0>,3.07<0,4,0>,4.07}
                     box{<20,3,20><-20,5,-20>}
                     pigment{Red}
                     scale<2.4,2.1,1.75>}
          }
```

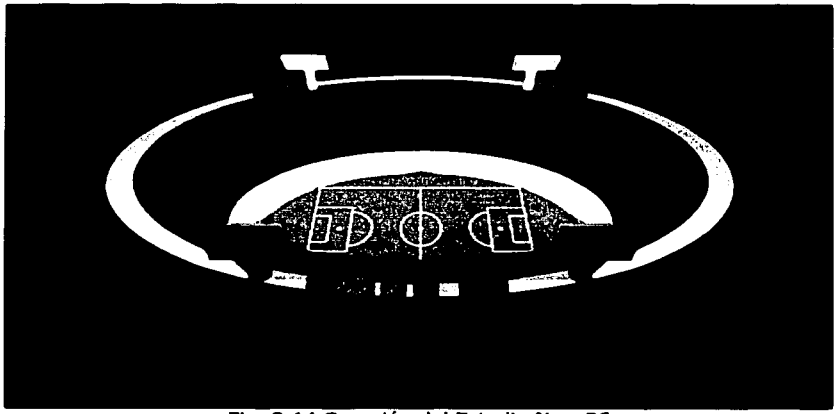


Fig. 2.14 Creación del Estadio Neza86

2.4. Modelos de Obtención de Información

Para realizar los modelos en 3D de los pozos, cárcamos, cochecito y tanque de regulación fueron proporcionadas las fotografías para que las utilizáramos como base para la construcción del modelo.

Se utilizo solamente Pov-Ray y Geometría Sólida Constructiva. A continuación se muestran las fotografías:

Fig. 2.15 Pozo de bombeo.

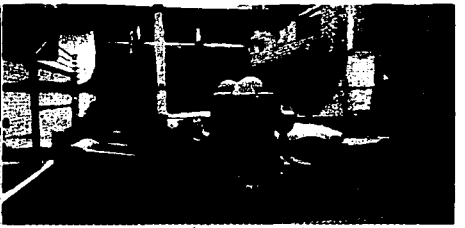


Fig. 2.16 Cárcamo de bombeo

Fig. 2.17 Tanque de regulación

Fig. 2.18 Vehículo para análisis

Ahora mostraremos el código que se necesita para la construcción de las figuras tomando como base las fotografías antes mostradas, así como algunas perspectivas del modelo.

2.4.1. Coche

```
// ********************

#include "colors.inc"
#include "textures.inc"

camera{location<0,0,18>
      look_at<0,-2,0>}

light_source{<0,0,20>color rgb<1,1,1>}
light_source{<4,6,10>color rgb<1,1,1>}
light_source{<-4,6,10>color rgb<1,1,1>}
light_source{<0,20,0>color rgb<1,1,1>}

// Creación del vehículo
#declare caja1=union{box{<5,-1.5,0> <-5,1.5,5>
                    texture{pigment{DMFWood4}}}
                union{difference{cylinder{<-3,0,5>,<-3,0,0>,2.5
                                    texture {pigment{DMFWood4}}}
                                box{<5,-1.2,0> <-5,-2.5,5.5>
                                   pigment { color rgb <0,0,0>}}
                                box{<5,1.4,0> <-5,2.5,5.5>
                                   pigment { color rgb <0,0,0>}}
                                box{<5,-1.4,0> <-5,-2.5,-1>
                                   pigment { color rgb <0,0,0>}}
                                cylinder{<-3,2.15,5.05>,<-3,2.15,-02>,.70}
                                }
                      cylinder{<-3,2.15,5.06>,<-3,2.15,-0.3>,.70
                            texture {Glass}}
                      }
                texture{pigment{DMFWood4}}}

//parte superior del auto
#declare caja2=union{difference{box{<2,1.4,0> <-5,3.5,5>
                              texture{pigment{DMFWood4}}}
                        cylinder{<-.3,2.15,5.06>,<-.3,2.15,-0.2>,.70
                              texture {Glass}}}
                cylinder{<2.2,3,5>,<2.2,3,0>,.5
                        texture{pigment{DMFWood4}}}
                cylinder{<-3,2.15,5.06>,<-3,2.15,-0.3>,.70
                        texture {Glass}}
               }

//focos
#declare focos=union{sphere{<0,0,0>,.3
                        translate<5,1.2,4.5>
                        pigment { color rgb <1,0,0>}}
                sphere{<0,0,0>,.3
                        translate<5,1.2,0.5>
                        pigment { color rgb <1,0,0>}}
               }

//llantas
#declare
llantas=union{difference{cylinder{<0,0,0>,<0,0,0.5>,1.50
                                texture{pigment{DMFWood4}}}
                        cylinder{<0,0,0>,<0,0,1>,1.2
                                pigment{color rgb <0,0,0>}}
                        }
              cylinder{<0,0,0>,<0,0,0.5>,0.30
                      texture{pigment{DMFWood2}}}
```

Fig. 2.19 Vista lateral Coche

Fig. 2.20 Vista trasera Coche

```
        cone{<0,0,0>,0.05  <0,0,0.6>,0.15
            texture{New_Brass}}
        rotate<0,0,clock*15>    }

//ARMADO DEL VEHICULO

#declare auto=union{object{caja1}
                    object{caja2}
                    object{focos}
                    object{llantas   translate<3.5,-1,5>}
                    object{llantas   translate<-3.5,-1,5>}
                    object{llantas    rotate<0,180,0>   translate<-3.5,-1,0>}
                    object{llantas    rotate<0,180,0>   translate<3.5,-1,0>}
                    object{llantas    rotate<0,-90,-2>   translate<-5.55,0,2.5>}
                    cylinder{<-3,2.15,5.05>,<-3,2.15,-0.2>,.70
                          texture {Glass}}
                    translate<-1,-2.45,0>
                   }
```

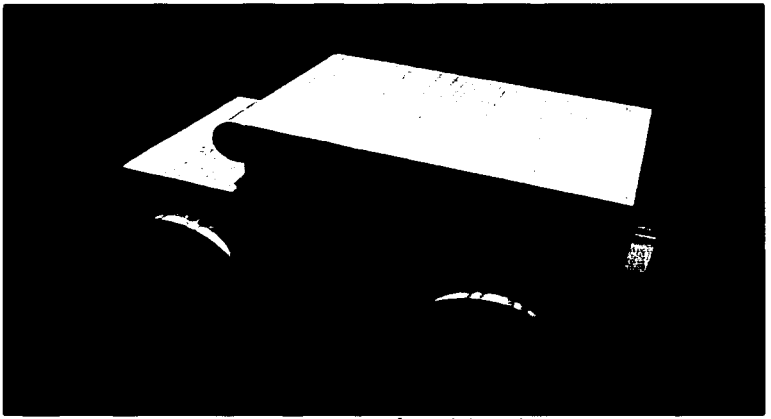


Fig. 2.21 Vista aérea del Coche

2.4.2. Cárcamo

```
// ************************************

#include "colors.inc"
#include "textures.inc"

camera{location<-2,6,23>      look_at<-2,5,5>}

light_source{<0,1,20>color rgb<1,1,1>}
light_source{<8,6,10>color rgb<1,1,1>}
light_source{<-8,6,10>color rgb<1,1,1>}
light_source{<0,20,0>color rgb<1,1,1>}

#declare rueda=union{torus{0.8,0.2}
                cylinder{<0,0,0.8>,<0,0,-0.8>,0.1}
                cylinder{<0.8,0,0>,<-0.8,0,0>,0.1}       }

#declare hoyos=union{box{<0.4,2.6,3><-0.4,3.4,-3>}
                box{<3,2.6,0.4><-3,3.4,-0.4>}
                pigment{Black}}

#declare bomba=difference{union{cylinder{<0,1,0>,<0,0,0>,1}
                        cylinder{<0,2.5,0>,<0,9,0>,2.5}
                        cylinder{<0,6.5,0>,<0,7,0>,3}
                        cylinder{<0,7.5,0>,<0,8.5,0>,3.2}
                        box{<-2.5,4.5,0.5><-4,6,-0.5>}
                        cone{<0,9,0>,2.5<0,10,0>,2}
                        cone{<0,10,0>,2<0,11,0>,1}
                        sphere{<0,11,0>1}}
                     object{hoyos}
                     object{hoyos     rotate<0,45,0>}
                     object{hoyos     translate<0,1,0>}
                     object{hoyos     rotate<0,45,0>   translate<0,1,0>}
                    }

#declare llave=union{cylinder{<-5,0,0>,<-5,2,0>,1}
                cylinder{<-5,1.9,0>,<-5,2.2,0>,1.2}
                sphere{<-5,2.2,0>1    scale<1,1.5,1>  translate<0,-1,0>}
                cylinder{<-5,3,0>,<-5,4.5,0>,.2}
                object{rueda   translate<-5,4.5,0>}
               }

#declare tubo_peque=union{cylinder{<6,0,0>,<-7.5,0,0>,1}
                     sphere{<6,0,0>1}
                     cylinder{<6,0,0>,<6,-18,0>,1}
                     object{llave     translate<7,0,0>}
                     cylinder{<0,0,0>,<-0.5,0,0>,1.4}
                     sphere{<-2,0,0>1.5}
                     cylinder{<-3.5,0,0>,<-4,0,0>,1.4}
                     cylinder{<-2,0,0>,<-2,2.2,0>,1}
                     cylinder{<-2,2,0>,<-2,2.5,0>,1.4}
                     texture {finish { Dull} pigment{Blue}}   }

#declare carcamo=union{box{<3.5,0,5><-5,5,-5>}
                  box{<3.55,3.5,5.05><-5.05,4,-5.05>  pigment{Blue}}
                  box{<3.7,5.2,5.2><-5.2,4.95,-5.2>}
                  cone{<-1.5,5.1,0>,2<-1.5,7,0>,1.5     pigment{Orange}}
                  object{bomba  scale<0.6,0.7,0.6>    translate<-1.5,5.25,0>
                        texture {finish { Dull} pigment{Blue}}  }
                  object{tubo_peque  scale<.5,.5,.5>  translate<3.5,6.25,0>}
                  pigment{Gray}
}
```

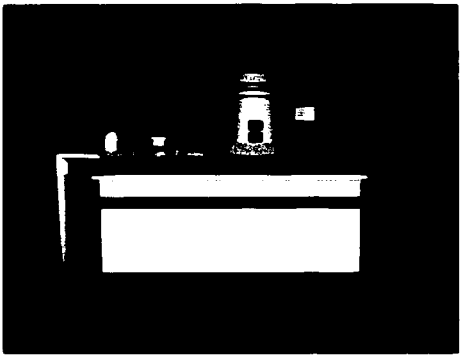


Fig. 2.22 Vista lateral Cárcamo

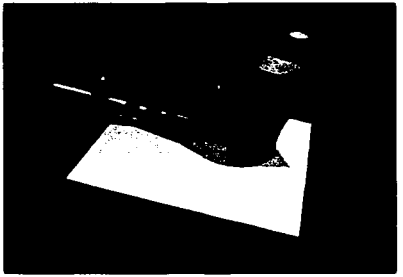


Fig. 2.23 Vista aérea Cárcamo

2.4.3. Pozo

```
// ********************************
```

```
#include "colors.inc"
#include "textures.inc"

#declare clo = clock;

 camera{location<-2,6,25>
          look_at<-2,4,5>}

light_source{<0,1,20>color rgb<1,1,1>}
light_source{<7,5,10>color rgb<1,1,1>}
light_source{<-7,5,10>color rgb<1,1,1>}


#declare llave=union{union{
                    cylinder{<0,7.5,0>,<0,9,0>,0.8}
                    sphere{<0,9,0>,0.8}
                    cylinder{<2,9,0>,<0,9,0>,0.8}
                    sphere{<2.7,9,0>,1.2}
                    cylinder{<3.5,9,0>,<5,9,0>,0.8}
                    cylinder{<2.7,10,0>,<2.7,11,0>,0.3}
                    texture{Gold_Texture}    }
                union{cylinder{<1.7,11.2,0>,<3.7,11.2,0>,0.4}
                    sphere{<1.5,11.2,0>,0.6}
                    sphere{<3.9,11.2,0>,0.6}
                    pigment{Red}}  }

#declare rueda=union{torus{0.8,0.2}
                cylinder{<0,0,0.8>,<0,0,-0.8>,0.17}
                cylinder{<0.8,0,0>,<-0.8,0,0>,0.17}  }

#declare hoyos=union{box{<0.4,2.6,3><-0.4,3.4,-3>}
                box{<3,2.6,0.4><-3,3.4,-0.4>}
                pigment{Black}}

#declare bomba=difference{union{cylinder{<0,1,0>,<0,0,0>,1}
                        cone{<0,1,0>,1<0,2.5,0>,2.5}
                        cylinder{<0,2.5,0>,<0,9,0>,2.5}
                        cone{<0,9,0>,2.5<0,10,0>,2}
                        cone{<0,10,0>,2<0,11,0>,1}
                        sphere{<0,11,0>1}}
                    object{hoyos}
                    object{hoyos  rotate<0,45,0>}
                    object{hoyos  translate<0,-1,0>}
                    object{hoyos   rotate<0,45,0>  translate<0,-1,0>}
                    object{hoyos   translate<0,1,0>}
                    object{hoyos   rotate<0,45,0>  translate<0,1,0>} }

#declare pozo=union{box{<7,2.5,5><-5,-3.5,-5>    pigment{Gray}}
                  box{<7.05,1,5.05><-5.05,1.5,-5.05>
                   pigment{Blue}}
                cylinder{<0,2.4,0>,<0,5.4,0>,0.8}
                sphere{<0,5.4,0>0.8}
                cylinder{<0,5.4,0>,<-5,5.4,0>,0.8}
                cylinder{<-3.5,6.1,0>,<-3.5,7,0>,0.15}
                cylinder{<-3.5,6.9,0>,<-3.5,8.35,0>,0.5}
                cylinder{<-3.5,7.9,0>,<-3.5,8.2,0>,0.65}
                cylinder{<-4.8,5.4,0>,<-6.8,5.4,0>,1.2}
```

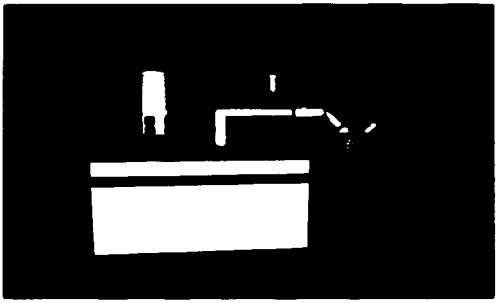


Fig. 2.23 Vista lateral Pozo

Fig. 2.24 Vista aérea Pozo

```
cylinder{<-6.7,5.4,0>,<-7.2,5.4,0>,0.8}
sphere{<-7.2,5.4,0>0.8}
cylinder{<-7.2,5.4,0>,<-17.4,-4.5,0>,0.8}
cylinder{<-8.2,4.4,0>,<-8.5,4.1,0>,1.2}
cylinder{<-9.2,2.8,0.8>,<-9.2,2.8,1.1>,1}
cylinder{<-11.2,1.4,0>,<-11.5,1.1,0>,1.2}
cylinder{<-13.2,-0.6,0>,<-13.5,-0.9,0>,1.2}

object{llave  scale<0.1,0.1,0.1>   rotate<0,90,0>  translate<-9.1,1.85,1.62>}

object{rueda   rotate<0,0,45>  translate<-11.63,5.05,0>}

cylinder{<-9.35,2.8,0>,<-10.85,4.3,0>,0.7}
cylinder{<-10.85,4.3,0>,<-10.55,4,0>,0.8}
sphere{<-10.85,4.3,0>0.7}
cylinder{<-10.85,4.3,0>,<-11.55,5,0>,0.25}

object{bomba   scale<0.7,0.7,0.7>   translate<4.5,2,0>}
texture {finish { Dull} pigment{Blue}}
}
```

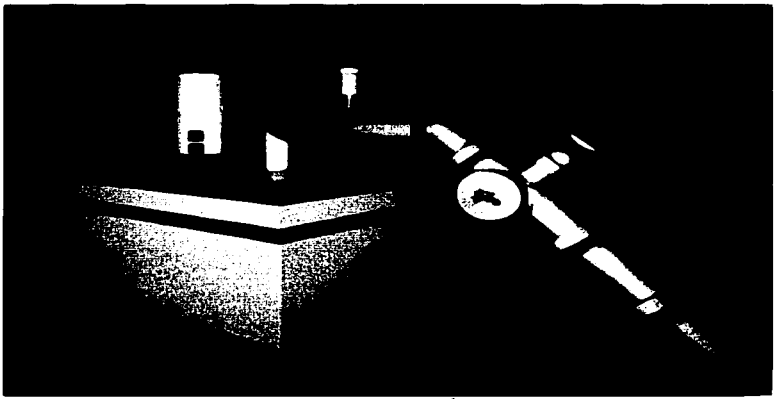


Fig. 2.25 Creación del Pozo

2.4.4. Tanque

```
// ********************************
```

```
#include "colors.inc"
#include "textures.inc"


#declare clo = clock;


camera{location<-2,4,25>
      look_at<-2,3.5,5>}

light_source{<0,1,20>color rgb<1,1,1>}
light_source{<7,5,10>color rgb<1,1,1>}
light_source{<-7,5,10>color rgb<1,1,1>}
light_source{<5,12,10>color rgb<1,1,1>}
light_source{<0,13,0>color rgb<0,1,1>}

#declare rueda=union{torus{.8,.2}
                    cylinder{<0,0,.8>,<0,0,-.8>,.1}
                    cylinder{<.8,0,0>,<-.8,0,0>,.1}
                   }

#declare codo=union{cylinder{<0,0,0>,<0,0,2>,1}
                   cylinder{<0,0,0>,<0,0,0.5>,1.2}
                   sphere{<0,0,2>1}
                   cylinder{<0,0,2>,<0,-2,2>,1}
                   cylinder{<0,-1.5,2>,<0,-2,2>,1.2}
                   scale<.2,.2,.2>
                   texture {finish { Dull} pigment{Blue}}
                  }

#declare codos=union{object{codo  translate<0,8,5>}
                     object{codo   translate<1.5,8,5>}
                     object{codo   translate<3,8,5>}
                     object{codo   translate<4.5,8,5>}
                     object{codo   translate<-1.5,8,5>}
                     object{codo   translate<-3,8,5>}
                     object{codo   translate<-4.5,8,5>}
                    }


#declare tubo_grande=union{cylinder{<0,-13,0>,<0,10,0>,1}
                          cylinder{<0,8,0>,<0,8.5,0>,1.2}
                          sphere{<0,10,0>1}
                          cylinder{<0,10,0>,<-9,10,0>,1}
                          cylinder{<-3,10,0>,<-3.5,10,0>,1.2}
                          cylinder{<-6,10,0>,<-6.5,10,0>,1.2}
                          sphere{<-9,10,0>1}
                          cylinder{<-9,10,0>,<-9,6,0>,1}
                          texture {finish { Dull} pigment{Blue}}
                         }


#declare tubo_peq=union{cylinder{<0,0,0>,<-11,0,0>,1}
                       cylinder{<-3,0,0>,<-3.5,0,0>,1.2}
                       cylinder{<-5,0,0>,<-5,2,0>,1}
                       cylinder{<-5,1.9,0>,<-5,2.2,0>,1.2}
                       sphere{<-5,2.2,0>1  scale<1,1.5,1>  translate<0,-1,0>}
                       cylinder{<-5,3,0>,<-5,4.5,0>,.2}
```

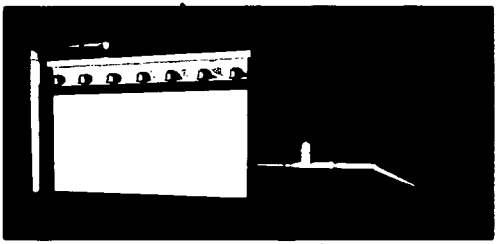


Fig. 2.26 Vista lateral de frente Tanque

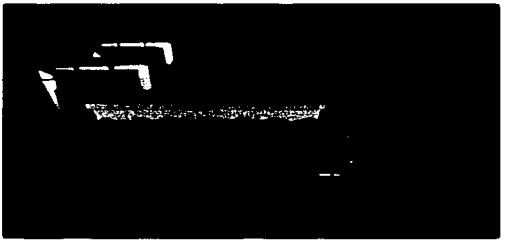


Fig. 2.27 Vista lateral aérea Tanque

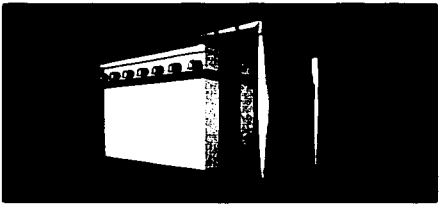


Fig. 2.28 Vista trasera Tanque

```
  object{rueda  translate<-5,4.5,0>}
  cylinder{<-7,0,0>,<-7.5,0,0>,1.2}
  sphere{<-11,0,0>1}
  cylinder{<-11,0,0>,<-20,-6,0>,1}
  texture {finish { Dull} pigment{Blue}}
 }

#declare tanque=union{box{<5,9,5><-5,0,-5>}
                box{<5.05,7,5.05><-5.05,7.5,-5.05>   pigment{Blue}}
                box{<5.2,8.9,5.2><-5.2,9.4,-5.2>}
                object{codos}
                object{codos   rotate<0,180,0>}
                object{tubo_grande   scale<.5,.5,.5>   translate<7.5,6,3>}
                object{tubo_grande   scale<.5,.5,.5>   translate<7.5,6,-3>}
                object{tubo_peq      scale<.5,.5,.5>  translate<-5,2,3>}
                object{tubo_peq      scale<.5,.5,.5>  translate<-5,2,-3>}

                pigment{Gray}
               }
```

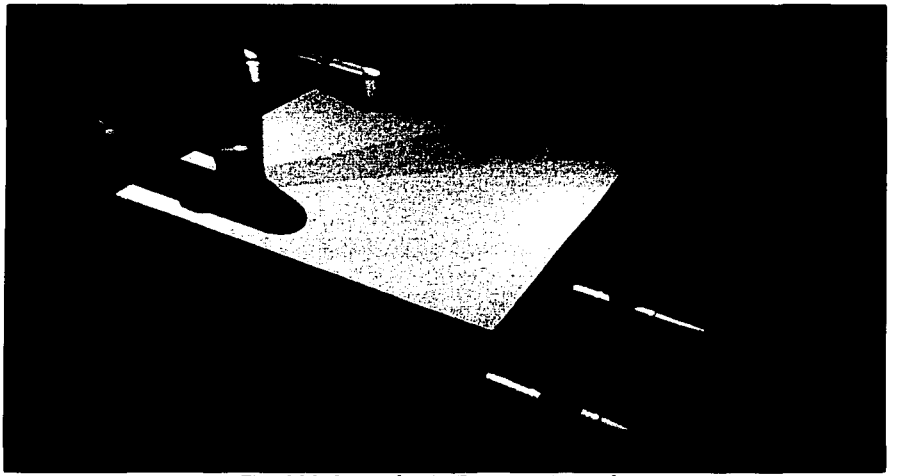


Fig. 229 Creación de Tanque vista aérea

2.5. Perspectivas del Plano

A continuación se muestran 6 perspectivas (figuras 2.30, 2.31, 2.32, 2.33, 2.34 y 2.35) de los diferentes ángulos de observación de la cámara sobre el plano 3D del proyecto con avenidas, nombres, manzanas, y simbología.

Fig. 2.30

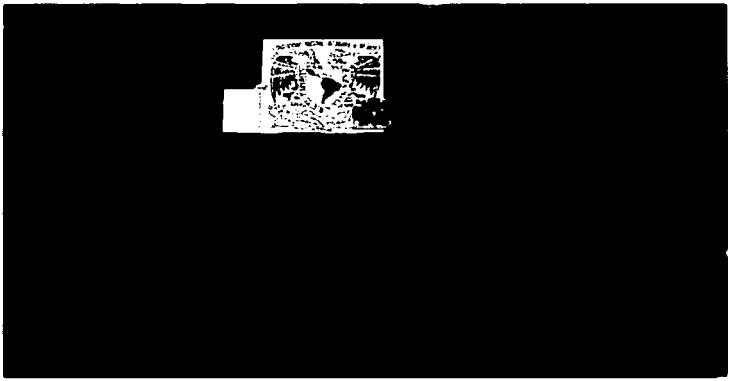


Fig. 2.31

Fig. 2.32

Fig. 2.33

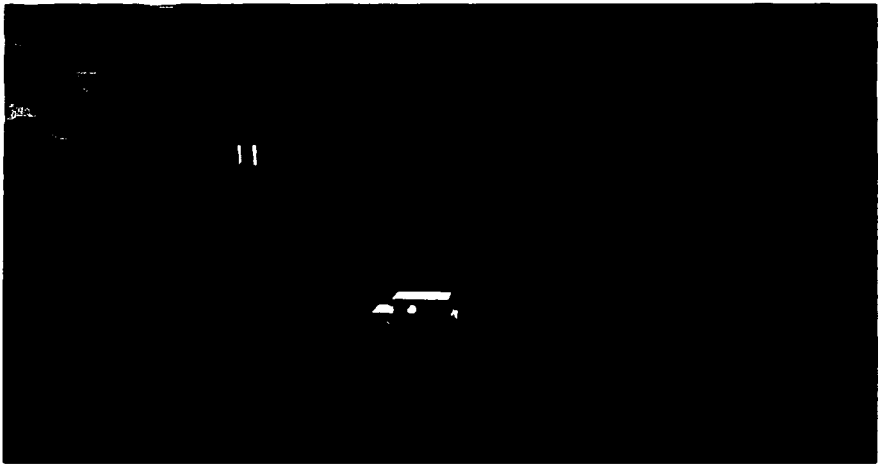


Fig. 2.34

Fig. 2.35

CAPÍTULO 3

Visión Gráfica de los Parámetros de Contaminación

3.1. Obtención de la Información

Los avances en la tecnología de la información y la graficación por computadora juegan hoy en día un papel muy importante en nuestras vidas, no hay área en la que no existan representaciones gráficas; científicos, personal médico, analistas comerciales y otros con frecuencia necesitan analizar grandes cantidades de información o estudiar el comportamiento de ciertos procesos, tan es así el caso del Laboratorio de Ingeniería Ambiental del CTA que acumulan archivos de datos mensuales de parámetros de contaminación de pozos y depósitos en Cd. Nezahualcóyotl que necesitan ser procesados y analizados, pero la obtención de estos conjuntos de números para determinar tendencias y relaciones es un proceso tedioso y poco eficiente. Pero si se convierten los datos a una forma visual y gráfica, es frecuente que se perciban de inmediato las tendencias y los patrones.

De los parámetros analizados por el laboratorio nosotros representamos los de mayor importancia, estos datos son tomados en cada uno de los pozos y depósitos, posteriormente son entregados en un archivo que contiene cierto formato para ser capturado por un programa elaborado en lenguaje "C" que se encarga de procesar la información y generar los archivos gráficos correspondientes.

3.2. Programa de Visualización en Lenguaje "C"

3.2.1. Entrada de Datos

Los Ingenieros del Laboratorio de Ingeniería Ambiental del Centro Tecnológico Aragón realizan un recorrido por Cd. Nezahualcóyotl en el cual recolectan una muestra de agua de cada uno de los pozos de bombeo, cárcamos de bombeo y tanque de regulación para ser analizada y procesada en el Centro Tecnológico Aragón; este análisis es llevado a cabo mensualmente obteniendo una serie de parámetros de contaminación que se encuentran en el agua.

Para este proyecto solo se toman en cuenta 11 parámetros, los cuales se enlistan:

PH
El término PH (Índice de Ion Hidrógeno) es usado universalmente para determinar si una solución es ácida o básica, es la forma de medir la concentración de iones de hidrógeno en la solución.

Cloro
El cloro C sirve en cantidades pequeñas en la desinfección del agua pero en cantidades mayores puede producir ciertos malestares.

Conductividad
La conductividad eléctrica es la capacidad que el agua tiene de conducir la corriente eléctrica. Este parámetro tiene relación con la existencia de iones disueltos en el agua, que son partículas con cargas eléctricas. Cuanto mayor sea la concentración de iones disueltos, mayor será la conductividad eléctrica en el agua.

Salinidad
Se entiende por salinidad a la concentración de sales disueltas en el agua.

Nitrato

Nitrato (NO_3) es un compuesto de nitrógeno y oxígeno y se encuentra en muchas de las comidas que se consumen diariamente. Aunque son bajos los niveles de nitrato que naturalmente ocurren en el agua, algunas veces se encuentran niveles altos que son muy peligrosos para infantes. Altos niveles de nitrato en el cuerpo pueden limitar la habilidad de la sangre de transportar oxígeno, causando asfixia en bebés. Esta condición podría ser fatal si no se trata a tiempo.

Nitrito

El Nitrito (NO_3 N) bajo ciertas condiciones puede ser muy peligroso, ya que reacciona con aminas formando compuestos cancerígenos.

Nitrógeno Amoniacal

El nitrógeno es uno de los elementos más importantes para la vida, pero es muy escaso en el agua. Sus fuentes principales son el aire (asimilado por algunas algas), adobos y materia orgánica en descomposición (hojas y aguas fecales). El nitrógeno que proviene de la descomposición de vegetales, animales y excrementos pasa por una serie de transformaciones. En el caso de los vegetales y animales, el nitrógeno se encuentra en forma orgánica. Al llegar al agua, es rápidamente transformado en nitrógeno amoniacal, pasando después para nitritos y finalmente a nitratos. Esas dos últimas transformaciones solamente ocurren en las aguas que contengan bastante oxígeno disuelto, pues son efectuadas por bacterias de naturaleza aerobia llamadas nitrobacterias. De esa forma, cuando encontramos mucho nitrógeno amoniacal en el agua, estamos en presencia de materiales orgánicos en descomposición y por lo tanto en un medio pobre en oxígeno.

Turbiedad

El agua puede ser turbia cuando recibe una determinada cantidad de partículas que permanecen algún tiempo en suspensión. Esto puede ocurrir como consecuencia de la lluvia que arrastra partículas de tierra hacia el río o como resultado de actividades del hombre tales como, minería (extracción de arena) y desagüe de residuos industriales.

Coliformes
Son bacterias producidas por heces fecales de animales y hombres que pueden desarrollar enfermedades en las personas.

Temperatura.
La temperatura es una variable muy importante en el medio acuático, pues influye en el metabolismo de las especies, como productividad primaria, respiración de los organismos y descomposición de la materia orgánica. Cuando tenemos altas temperaturas se produce una proliferación de fitoplancton, y, por consiguiente, intensa absorción de nutrientes disueltos. En caso de disminución de la temperatura se produce el efecto contrario.

Después de procesada la información por el Laboratorio de Ingeniería Ambiental del CTA, se nos proporciona un disco magnético de 3½" con un archivo con los datos obtenidos que tienen el siguiente formato:

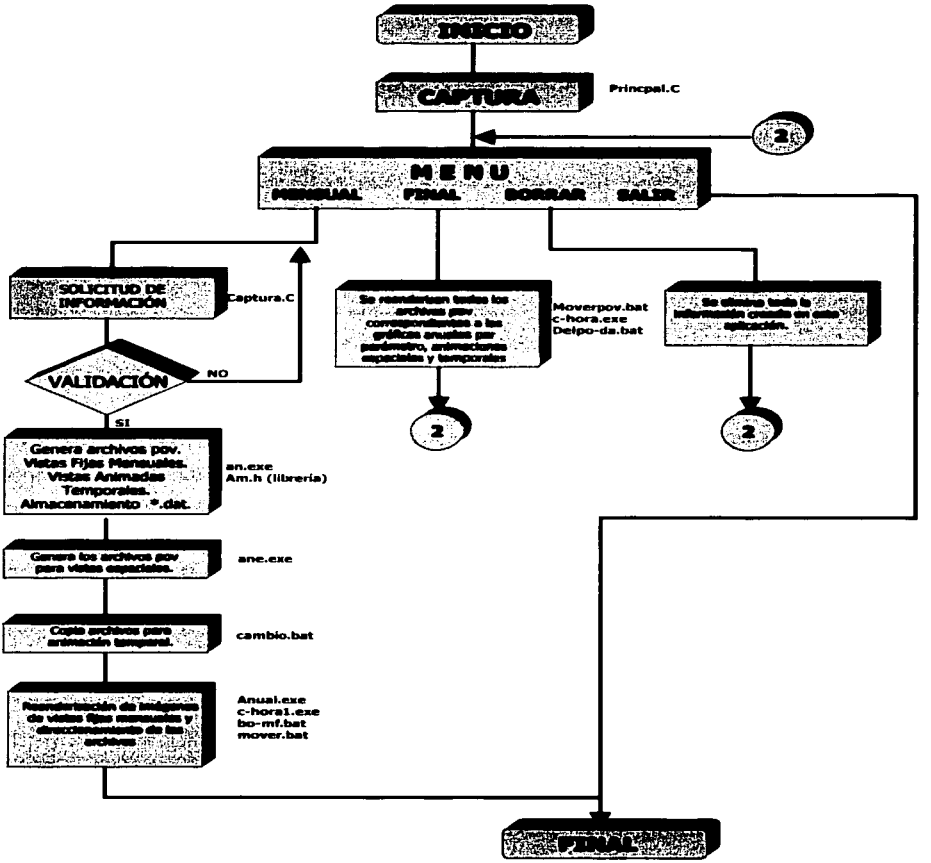
- El archivo debe ser de tipo texto *.txt
- Debe de tener el nombre del mes al que se hace referencia. Ejemplo. enero.txt
- Los parámetros a visualizar se proporcionan en forma de lista empezando por el nombre del sitio de acopio, y enseguida los 11 parámetros correspondientes en valor numérico de los antes citados.

Ejemplo.
<Nombre de centro de recopilación>
<Parámetro 1>
...
<Parámetro 11>

El archivo de texto es captado en un programa de lenguaje "C" que va recopilando la información, generando archivos de imágenes tridimensionales en forma mensual y anual muy significativas para la comprensión y toma de decisiones en el comportamiento de los parámetros de contaminación. Es muy importante seguir las características marcadas para el archivo para evitar generación de imágenes erróneas.

3.2.2. Procesamiento

A continuación se muestra el diagrama general del funcionamiento del programa:



Este proceso es generado como ya lo mencionamos utilizando un lenguaje de programación que en este caso es Lenguaje "C", y además nos apoyamos también en archivos de procesamiento por lotes.

El programa principal esta conformado por varios subprogramas, librerías y archivos que logran realizar de una manera óptima la creación de las imágenes o archivos gráficos.

Como primer paso es el de recibir el diskette de 3½" que contiene un archivo con formato de texto con el nombre correspondiente al mes de análisis.

Principal.c

El programa principal *"Principal.c"* se encarga de dar la presentación sobre el proyecto, pide una contraseña de validación para el acceso y muestra el menú principal para iniciar el proceso.

Fig. 3.1

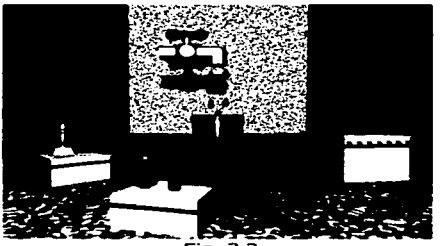


Fig. 3.2

Presentación del programa Principal.c

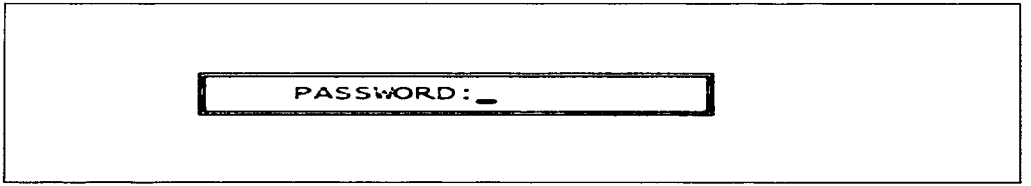


Fig. 3.3 Clave de acceso al sistema

El menú esta compuesto por las opciones:

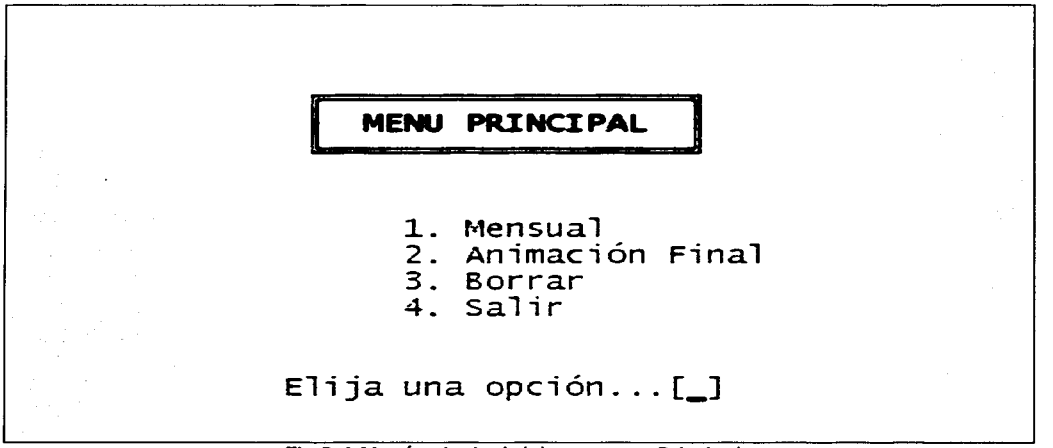


Fig.3.4 Menú principal del programa Principal.c

1. Mensual

Generará las vistas fijas mensuales, archivos para animaciones temporales y espaciales, así como almacenar los parámetros por mes para las vistas fijas anuales por parámetro.

2. Animación Final

En esta opción tendremos las gráficas de las vistas fijas anuales por parámetro y las animaciones espaciales y temporales.

3. Borrar

Borrará la información para un nuevo análisis al inicio del ciclo anual.

4. Salir

Ahora analizaremos los procesos a los que se refieren cada una de las opciones. Para esto se comenzará con la opción número uno que se refiere a la mensual.

Mensual

Después de haber elegido esta opción se manda llamar un subprograma también realizado en lenguaje "C" que se llama "*Captura.exe*" en el cual nos muestra un nuevo menú que nos indica los meses del año para poder captar la información del archivo de texto.

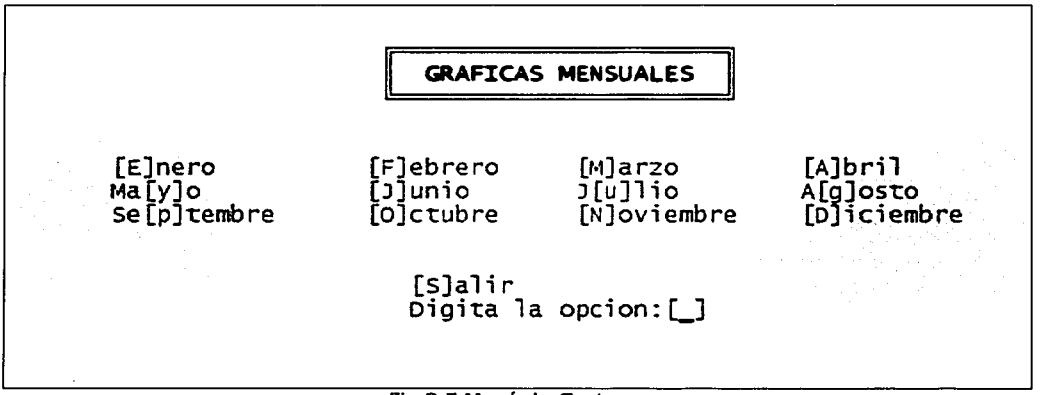


Fig.3.5 Menú de Captura.c

Al elegir alguna de las opciones se valida el mes al que se hace referencia, esto implica que se mandan llamar 2 subprogramas en lenguaje "C" los cuales son An*.exe y Ane*.exe, además de un archivo por lotes Cambio*.bat que es el encargado de redireccionar los archivos a una carpeta específica que es donde se van almacenar todos los datos. El * representa un valor numérico con relación al mes.

Con estos subprogramas se captan los datos y se generan los archivos gráficos para las diversas vistas ya mencionadas, así como el almacenamiento para datos futuros, estos

programas son auxiliados con archivos de texto y una librería que tienen partes elementales predefinidas que van conformando lo que son nuestros archivos de salida.

Los programas An*.exe nos generan los archivos de vistas fijas mensuales y animaciones temporales, este programa se complementa con una librería Am.h que almacena y guarda los datos para la generación posterior de las gráficas anuales por parámetro en archivos con extensión dat, mientras que Ane*.exe nos proporcionará las vistas animadas espaciales.

Estos archivos de texto son Cabecera.txt, Cabecera2.txt, Color.txt, Color1.txt, Nombre.txt, Nombre1.txt, Nombre2.txt, Tipodeletra.txt.

Se van abriendo los archivos de texto comenzando con la cabecera, agregando los nombres de cada lugar de acopio, un color referente a cada parámetro, y se insertan los valores del archivo de texto entregado a este proyecto por el CTA para poder dar como resultado los archivos con extensión pov referentes a todas las vistas gráficas que se manejan en las animaciones espaciales, vistas fijas mensuales y almacenamiento de los archivos dat.

Una vez teniendo todos los datos se ejecuta el programa Anual.exe que a su vez manda llamar a los archivos siguientes:

Gene.txt que es en el que contiene la lista de los archivos pov referentes a las vistas anuales por parámetro.
Gene1.txt son los archivos dat que tienen los valores de cada una de la barras de cada parámetro.
Gene2.txt contiene la cabecera específica de cada parámetro.

Esto con la finalidad de complementar la información contenida en los archivos pov dando un total de 110 archivos.

De estos archivos se toman 11 que son los que se van a procesar para generar las 11 vistas fijas mensuales utilizando el programa c_hora1.exe que se vincula con el programa Pov-Ray. Los demás archivos son almacenados para el procesamiento anual.

Para el redireccionamiento de todos los archivos generados se utilizan los archivos mover.bat y bo-mf.bat que son los encargados de mover todos los archivos con extensión dat y pov, ya que cuando se termina el primer procesamiento de un mes en específico al iniciar otro si no se han movido serán remplazados. Cabe mencionar que este proceso se llevará a cabo cada mes.

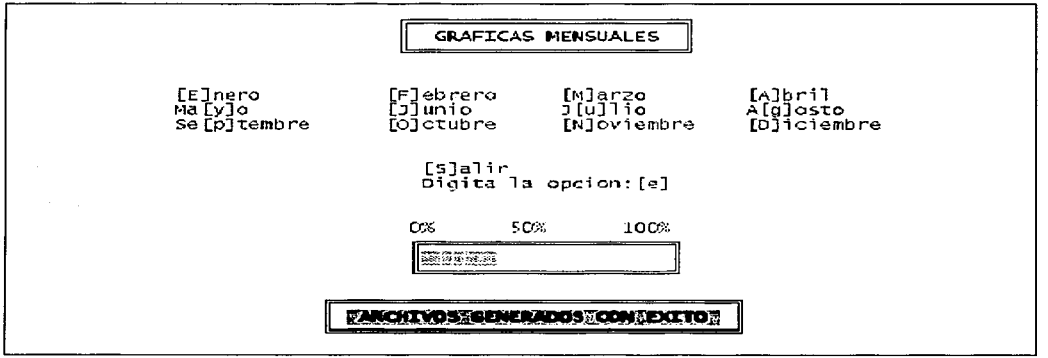


Fig. 3.6 Elección del menú para captura de información por mes. Programa Captura.c

Anual

En esta opción del menú principal se ejecutan los programas moverpov.bat, delpo-da.bat que servirán como redireccionamiento de los archivos pov y de mover los archivos a las carpetas correspondientes, posteriormente el programa c-hora.exe realiza la renderización de todos los archivos generados en el año con extensión pov dándonos como resultado archivos en formato de imagen bmp. Para poder realizar la animación es necesario convertir los archivos bmp en formato jpeg, para lo cual utilizaremos el programa de software libre Lview-Pro.

Teniendo las imágenes en formato jpeg se utiliza el programa Vfd para realizar las diferentes animaciones temporales, espaciales y anuales.

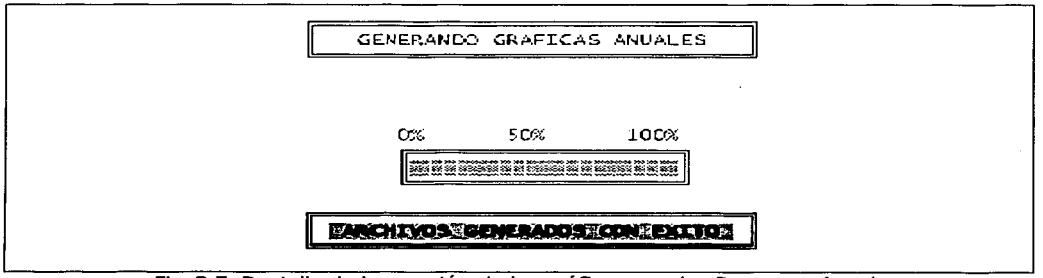


Fig. 3.7 Pantalla de la creación de las gráficas anuales Programa Anual.c

Borrar

En este punto se va a eliminar toda la información contenida en las carpetas utilizadas en la creación y procesamiento de las imágenes. Ésta es una opción muy importante ya que si se selecciona por equivocación perderá todos los datos, por lo que se consideró poner una contraseña de acceso para la eliminación de archivos.

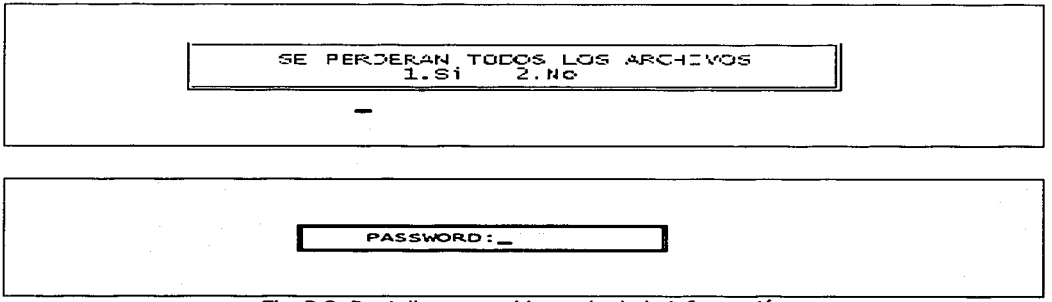


Fig. 3.8 Pantallas para el borrado de la información

3.2.3. Salida de Datos por Visualización

La información obtenida, analizada, capturada y procesada por el laboratorio de Ingeniería Ambiental y por este proyecto de tesis sobre los parámetros de contaminación es muy importante visualizarla de una manera fácil y práctica para poder aprovechar al máximo toda la información, por este motivo utilizamos los archivos de imagen generados en forma de gráfica de barras por el programa de lenguaje "C" y Pov-Ray. Los archivos generados se clasifican en:

- Vistas fijas: mensuales y por parámetro
- Vistas animadas: temporales y espaciales.

Estas visualizaciones gráficas contemplan los parámetros antes ya mencionados y se representa una gráfica por lugar de muestreo.

Vistas fijas mensuales. Muestra una comparación entre el valor obtenido del análisis del lugar de acopio contra el límite permisible por la Norma Oficial Mexicana NOM-127-SSA1-1994. (Gráficas realizadas 11)

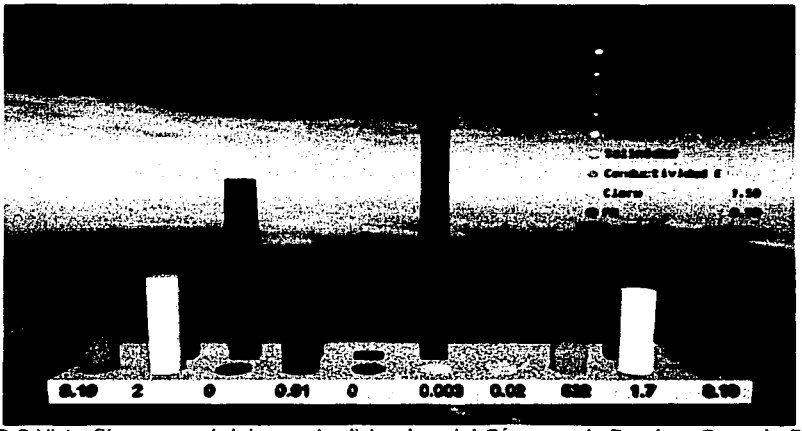


Fig. 3.9 Vista fija mensual del mes de diciembre del Cárcamo de Bombeo Carmelo Pérez

Vistas fijas por parámetros. Aquí conocemos como es el comportamiento y seguimiento de un parámetro específico durante un año obteniendo el seguimiento mensual.

(Gráficas 110)

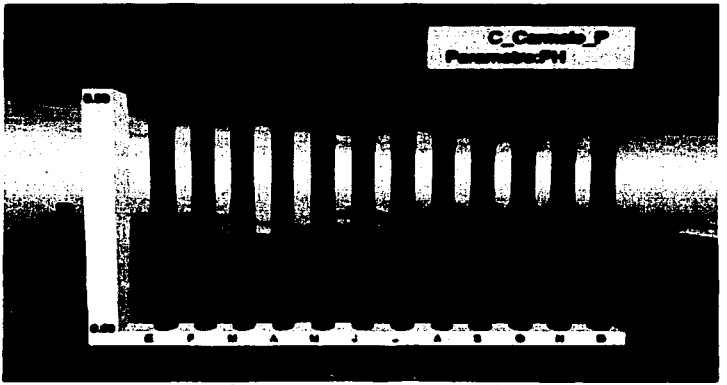


Fig. 3.10 Vista por parámetro durante todo un año del comportamiento del PH en el pozo 303

Animación Temporal. Proporciona el comportamiento de los parámetros de todo el año en forma animada, contraponiendo el pozo contra el mes. Similar a la vista fija mensual.

(Gráficas 132, 11 animaciones)

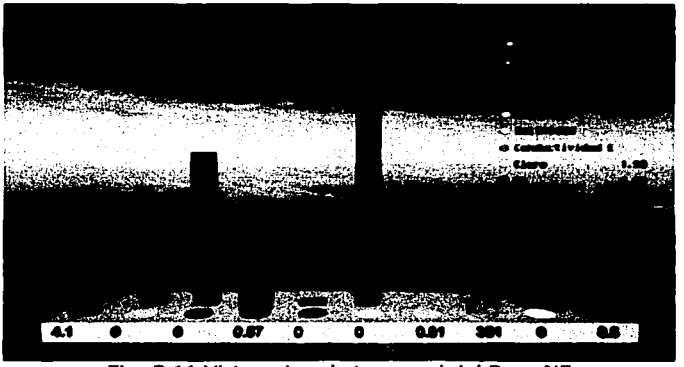


Fig. 3.11 Vista animada temporal del Pozo N5

Animación Espacial. Muestra el comportamiento de los parámetros de todo el año de manera animada, contraponiendo el mes contra el pozo, así como indicando la ubicación de los lugares de muestreo. (Gráficas 132, 12 animaciones)

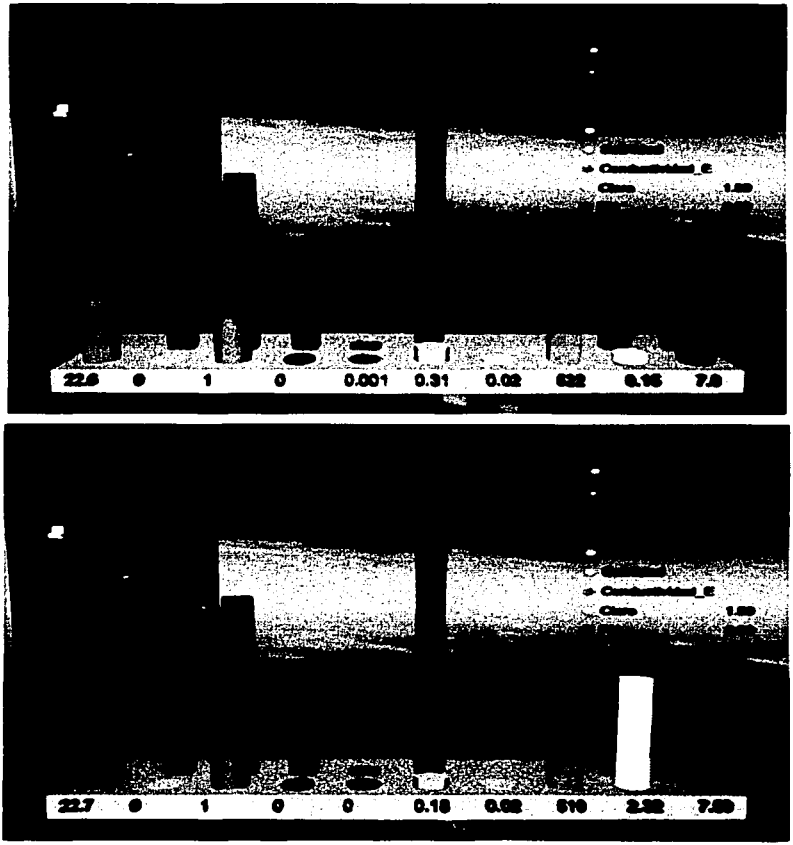
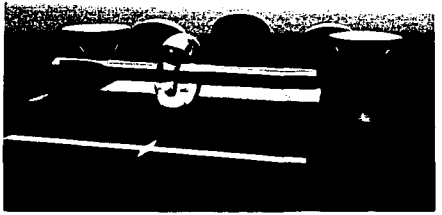


Fig. 3.12 y Fig. 3.13 Vistas animadas del cárcamo de bombeo Carmelo Pérez

CAPÍTULO 4

Animaciones



4.1. Antecedentes

Animación es la creación de la ilusión de movimiento al ver una sucesión de imágenes fijas generadas por el ordenador. Antes de la llegada de las computadoras, la animación se realizaba filmando secuencias dibujadas o pintadas manualmente sobre plástico o papel, denominados celuloides o fotograma una a la vez. Al principio, las computadoras se utilizaron para controlar los movimientos de la obra artística y simular la cámara.

La animación informática puede utilizarse para crear efectos especiales y para simular imágenes imposibles de generar con otras técnicas. La animación puede generar imágenes para datos científicos, y se ha utilizado para visualizar grandes cantidades de datos en el estudio de las interacciones de sistemas complejos como: la dinámica de fluidos, las colisiones de partículas y la simulación de tormentas, etc. Estos modelos de base matemática utilizan la animación para ayudar a los investigadores a visualizarlas.

Nosotros empleamos estas animaciones para facilitar la investigación y la toma de decisiones, mostrando los resultados más agradables a la vista y con una interpretación más fácil.

Para poder desarrollar una animación es necesario considerar que todos hemos observado a través de la televisión, lo que es la apariencia del movimiento continuo a partir de un numero de imágenes determinadas por segundo, este proceso se le llama persistencia de la visión.

Este proceso se realiza cuando el ojo humano recibe constantemente información visual, pero no es de una manera continua, más bien toma imágenes fijas en cada intervalo de tiempo, en el cerebro se crea la secuencia de forma que parece un flujo constante de imágenes, sin necesidad de procesar cantidades gigantescas de información visual.

La cantidad de imágenes que podemos percibir en una forma constante por segundo es de 12 a 16 cuadros; si queremos mostrar una animación debemos de contar con una cantidad determinadas de imágenes por segundo. El cine se estandariza en 24 cuadros por segundo mientras que la televisión trabaja con 30 cuadros por segundo. El realizar una animación con un mayor número de cuadros no implica que mejore la calidad, esto es debido a la percepción que tiene el ojo humano, en cambio menos cuadros por segundo tendremos una visión más lenta y con más de 30 no observaríamos cambio alguno, esto sólo provocará que se gasten recursos innecesarios en la creación de una animación.

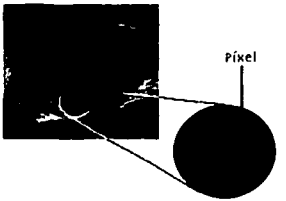
Dentro de esté capítulo es necesario conocer algunos términos que son muy importantes en el empleo de la animación como son:

Píxel

Término en inglés "picture element". Se trata de un punto en una rejilla rectilínea de miles de puntos tratados individualmente, para formar una imagen en la pantalla de la computadora o en la impresora. Igual que un bit es la unidad de información más pequeña que puede procesar un ordenador o computadora, un píxel es el elemento más pequeño que el *hardware* y el *software* pueden manipular al crear cartas, números o gráficos. Por ejemplo, la letra A está compuesta realmente por un conjunto de píxeles dentro de una rejilla como la que se muestra a continuación.

Una imagen también se puede representar con más de dos colores. Si un píxel tiene sólo dos valores de color (normalmente blanco y negro), se puede codificar con un solo bit de información. Cuando se utilizan más de dos bits para representar un píxel, es posible representar un rango mayor de colores y niveles de gris. Con dos bits se representan cuatro colores o diferentes tonos de color gris, con cuatro bits se representan dieciséis colores, y así sucesivamente.

Estos puntos o bloques tienen cada uno su color para mostrar imágenes en las pantallas del ordenador o computadora, y representan los elementos más pequeños que pueden manipularse para generar gráficos. Como no son infinitamente pequeños, los píxeles sólo se aproximan al color real de un objeto. Por esta razón, las líneas generadas por los gráficos de una computadora (denominadas mapas de bits) suelen tener un aspecto dentado si se observan de cerca.



Resolución

La capacidad de video de un ordenador o computadora se define como el número de píxeles por unidad de medida, se utiliza generalmente para indicar el número de píxeles mostrados horizontal o verticalmente en el monitor de video.
Las resoluciones más utilizadas son:

320 x 240 píxeles
640 x 480 píxeles
800 x 600 píxeles
1024 x 720 píxeles

Entre más grande sea la resolución, la calidad de la imagen será mucho mejor, pero el espacio utilizado en el disco se incrementará haciendo un poco difícil su procesamiento en procesos posteriores.

Cuadro.
Es una imagen fija e individual que forma parte de una animación, es conocido como fotograma en los medios cineastas.

Cuadros por segundo
Indican la cantidad de imágenes individuales que se muestran en un segundo para la creación de una animación.

Paleta de Colores
Es el conjunto de colores formados por la unión de varios bits, es decir para formar una paleta de cuatro colores la formamos con 2 bits codificándolos y asignándoles un color específico de la siguiente manera:

00	Color 1
01	Color 2
10	Color 3
11	Color 4

Si queremos una paleta de colores más amplia, solo tenemos que trabajar con más bits al mismo tiempo, veamos la siguiente tabla:

Bits	Paleta de Colores
2	4
4	16
8	256
16	65536
24	16777216

Una imagen con una paleta de colores de 16 ó 24 bits se asemeja al color real, sólo que el incremento de espacio en el almacenamiento es superior con relación a una imagen que contenga menos colores y al mismo tiempo el proceso sobre una imagen de calidad real hace la necesidad de tener hardware y software muy potente para minimizar el tiempo de proceso.

Cámara
Toma la posesión imaginaria del observador para la creación de las escenas y poder formar la animación.

4.2. Recursos de Animación

Las animaciones por computadora en nuestra actualidad están siendo muy utilizadas en todos los campos de estudio, desde aplicaciones médicas y otras ramas afines, investigación, mercadotecnia, cinematografía, etc. El problema al que nos hemos enfrentado desde el inicio de la creación de animaciones es que los recursos que se necesitan tanto financieros como físicos son muy grandes, a lo largo de estos años se ha ido dando solución a este problema, pero todavía falta un poco más de tiempo para que sea más accesible para todas las personas. Para el desarrollo de las animaciones debemos de tomar en cuenta los siguientes recursos:

Recurso de Almacenamiento.
Se refiere a tener un medio físico de almacenamiento muy grande. Esto va a depender de diferentes factores como son: a) La resolución con la que se reproduzcan las imágenes que pueden ser de 320 x 240, 640 x 480, etc. b) La cantidad de cuadros por segundo a la que se desarrolle. c) La paleta de colores que contengan las imágenes, ya que ésta influye en la cantidad de píxeles y bits, por que entre más calidad se le aplique a la imagen esta se ira duplicando o triplicando como observamos en la tabla de resolución; y por último

adicionando los diferentes formatos de audio que se utilicen para incorporar el sonido que dará por terminada la animación.

Recurso de Procesamiento.

Este recurso es de gran importancia ya que es el encargado de minimizar el tiempo de generación de las imágenes individuales y, después, en conjunto para formar las animaciones.

Los píxeles son un punto a tratar muy importante, ya que debemos de realizar muchas operaciones sobre un solo píxel para poder definir que estructura de color va a contener.

Para calcular un píxel es necesario hacer varias operaciones simples, esto implica que al multiplicar todas estas operaciones por el total de puntos que forman la imagen resultarán millones de operaciones, lo que provoca que la renderización de la imagen dependa principalmente del poder de cálculo que tienen las computadoras. Con el crecimiento de éstas en la actualidad se está permitiendo reducir los tiempos de generación de las imágenes, ya que la cantidad de operaciones se procesa a una velocidad mucho más rápida.

Ahora en la actualidad con el desarrollo de plataformas tipo servidor que tienen un potencial muy alto de trabajo, el proceso de creación de animaciones se está haciendo más fácil y accesible, se difunde con mayor facilidad en todos los ámbitos de la sociedad.

Las computadoras personales ya cuentan con un poder de cálculo suficiente para poder hacer animaciones pequeñas en casa de buena calidad.

Los tiempo de generación de una imagen procesada en una plataforma tipo servidor varía considerablemente en el tiempo de proceso de una PC, esto lo vemos en la siguiente tabla comparativa.

Plataforma	Tiempo
PC 486 DX2/66 MHz	5 Hrs. 55 min.
PC Pentium 60 MHz	1 Hr. 45 mim.
Sun Sparck Classic	30 mim
Sun Spark 4	12 mim 50 seg.
Silicon Grafhics 02	2 mim 39 seg.

Archivo chess.pov de Pov-Ray versión 2.0 a una resolución de 800 x 600. [4.1]

Recurso de Despliegue

Casi cualquier computadora tiene la capacidad de almacenar los cientos de MegaBytes de información de las imágenes individuales (cuadros), pero faltaría hacer el proceso de unir todos estos cuadros por segundo para formar la animación. Este proceso es muy complicado ya que es necesario que exista una interacción de mucha rapidez entre el procesador y una gran capacidad de memoria para que se realicen las dos actividades simultáneamente, porque uno está procesando mientras la otra está haciendo la función de despliegue de la animación en tiempo real.

4.3. Programa VFD

VFD es un programa de libre distribución para el sistema operativo MS-DOS que crea archivos de animación con extensiones: *.FLC, *.FLI y *.AVI tomando archivos base como:

Archivos Windows	*.RDI	
Archivos Windows	*.BMP	de: 1, 4, 8, 16 y 24 bits
Archivos Windows	*.RLE	de: 4 y 8 bits
Archivos IBM OS/2	*.BMP	de: 1, 4, 8 y 24 bits
Archivos Truevision	*.TGA	de: 8, 16, 24 y 32 bits
Archivos Zsoft	*.PCX	de: 8 y 24 bits
Archivos Autodesk	*.FLC, **.FLI	
Archivos Windows	*.AVI	de: *.RLE y 8, 16, y 24 bits

* Se refiere al nombre del archivo

[4.1] Tesis "Generación de Animaciones" Autor. Ing. Marcelo Pérez Medel. México D.F. 8 – Dic. -1997.

El programa VFD utiliza un avanzado método de cuantización de color llamado Hilite que produce video en movimiento de 8 bits con una gran fidelidad de imagen y color, a partir de un diverso rango de archivos de imágenes y paletas; empleando cuantización escalable de una reducción de color y difuminado de intensidad variable.

Lo podemos emplear para crear una paleta óptima que contenga los colores más comunes de los archivo de entrada.

Parámetros para un Archivo Animado de Salida

Se escribe el nombre del archivo de entrada seguidamente del parámetro, deben estar separados por espacios y se identifican por medio de un guión (-).

La sintaxis para generar un archivo animado de video es: pasar a modo MS-Dos donde se escriben las iniciales del programa VFD, enseguida el nombre del archivo de entrada secuenciado y seguidamente los parámetros que se enlistan:

VFD archivo de entrada [parámetros]

-An	Crea un archivo cuando n=1 de 8 Bit Crea un archivo cuando n=2		*.AVI *.RLE
-I	Crea un archivo **.FLI que por defecto es		*.FLC
-Mn	Compresión n=1 50%, n=4 80%	n=2 67%, ,...,	n=3 75% n=99 99%
-n=4	8-bit *.PCX, n = 5, 6, 7: 8, 16, 24 bits	*.TGA sin compresión	
-Sn	Velocidad de reproducción en cuadros/segundo n=0.1 a 99		
-Ofname	Ruta de salida		
-Tn1[,n2]	Total de cuadros		

Una de las partes más importantes de este programa además de poder leer cualquier archivo de los antes citados, es la posibilidad de emplear comodines (* ó ?) para poder identificar las series sucesivas de imágenes para darle forma a la animación y utilizando los parámetros del programa VFD ya mencionados para poder configurar de la manera más apropiada dicha animación.

Existe una limitante en la creación de animaciones la cual marca que no podemos exceder de 65535 cuadros sucesivos.

El programa VFD puede leer archivos en formato de texto con una extensión *.mak, trabaja como un guión de una escena de película, esto es realizado por medio de un procesamiento por lotes donde se van incluyendo los procesos para ir dando formato a la animación. Para está unión de procesos se utiliza un punto y coma (;) para los comentarios sin interferir en el proceso. Este programa nos marca la limitante de que los archivos que se utilicen para formar el guión no pueden exceder de un tamaño de 16 Kb.

Al ir construyendo nuestras animaciones nos encontramos que los archivos de imágenes contienen una gama de colores que van de 1 a 32 bits que provoca que aumente en gran tamaño el archivo. El programa tiene un contador que cuando se excede de 256 colores utiliza una técnica de suavizados para obtener la salida óptima para el video de 8 bits. VFD construye un histograma de frecuencia de todos los colores de una imagen de entrada para formar una propia paleta de colores más frecuentes; se utiliza el método Hilite que consiste en la reducción de colores el cual produce resultados mejores o similares a los métodos estándares.

La Compresión (-Mn)

El programa VFD nos proporciona dos formas de comprimir el video: 1) Reduciendo la calidad de la imagen de cada cuadro y 2) Reduciendo el número de cuadros por segundo.

El Redimensionamiento (-RI)

El parámetro permite redimensionar el tamaño vertical y horizontal de la imagen de entrada, es decir cambiar su tamaño de: 640 x 480 → 320 x 240 píxeles.

En el redimensionamiento existe la limitante que los archivos de imagen de entrada no deben exceder los 1024 x 768 píxeles.

La Velocidad

Este parámetro controla la velocidad de reproducción que puede variar desde un rango de 0.1 a 99 cuadros por segundo, esto se determina por el parámetro de la siguiente forma:

-S.1	10 segundos por cuadro
-S.2	5 segundos por cuadro
-S10	10 cuadros por segundo

El Audio

Para integrar el sonido en los archivos de animación de salida, el programa VFD salta sobre las pistas de audio de los archivos de entrada. Aquí es necesario auxiliarnos de los programas Video For Windows 'VidEdit' y 'WavEdit' que los podemos utilizar para insertar el audio a nuestros archivos creados.

Requerimientos del Sistema

Este programa necesita de especificaciones mínimas para poder obtener resultados óptimos:

- 500 Kb de memoria convencional y 96 Kb de memoria XMS para procesar imágenes de 320 x 200 para imágenes de mayor tamaño se requiere de contar con más memoria.
- Sistema operativo como mínimo MS-DOS 3.1, Windows 3.11 o superior.
- Procesador 80286 o superior.
- Una tarjeta de video SVGA requerida para reproducir videos a 256 colores.

4.4. Formatos de Animación

El Formato GIF

Fue creado por CompuServe con el objetivo de proporcionar un camino para intercambiar archivos de imagen de una forma rápida a través de la línea telefónica. Los archivos GIF se almacenan en un formato comprimido, de tal manera que el tiempo que se emplea para cargar estos archivos gráficos es mínimo. Los archivos GIF soportan tipos de imágenes de color indexado, así como imágenes en escala de grises y de líneas. Una ventaja muy importante de los archivos GIF es que pueden mostrarse en las plataformas UNÍX, Mac y Windows.

Hay dos tipos de formatos de archivos GIF: GIF89a y GIF87a. GIF89a incorpora un índice de transparencias que hace que el color de fondo permanezca inalterado para el color indexado que se haya seleccionado como transparente. Las imágenes entrelazadas permiten que se muestren de manera progresiva. Cuando se carga un archivo entrelazado, las imágenes aparecerán con un efecto de persiana veneciana, pero permiten que el usuario comience a ver el resto del documento mientras la imagen GIF aún se está cargando. Los archivos GIF están soportados tanto por la versión 2.0 de NCSA Mosaic como por Netscape Navigator.

El Formato JPEG

JPEG es el estándar a elegir, debido a su alta resolución y a su elevada compresión. Muchos editores gráficos, tal como Adobe PhotoShop, permiten elegir una configuración de alta calidad para efectuar la compresión. La elevada calidad se comprime menos, con una relación comprendida entre 5:1 y 15:1. Las imágenes JPEG se descomprimen automáticamente cuando se cargan en el navegador. JPEG reduce los archivos de imagen aproximadamente a un 10% de su tamaño original. El algoritmo JPEG pierde algunos de los datos, ya que identifica e ignora los píxeles que no son esenciales para la calidad general de la imagen; por ejemplo una gran área de un color continúo. Típicamente, la ausencia de la información sustraída no es observable. Pero después de comprimir una

imagen utilizando JPEG se pierde alguna información, por lo que la imagen puede no ser distinguible del archivo JPEG original, y además será mucho menor. Otra ventaja de JPEG es que está soportado en el Nivel 2 de PostScript. Por su parte, PostScrip ha sido desde hace mucho tiempo un estándar en Internet. Cuando una imagen comprimida con JPEG se envía a una impresora PostScrip de Nivel 2, el archivo se envía a la impresora y después se descomprime. A diferencia de lo que sucede con GIF, JPEG que soporta colores hasta de 24 bits.

Publicación de Video

El video ha tenido un crecimiento acelerado, de modo que está entrando ya en su adolescencia. En su estado actual, el video todavía tiene muchos problemas. No existe una normativa clara y es incómodo de manejar, pero existen grandes esperanzas para este mundo de la imagen en movimiento. Los apartados siguientes muestran la compresión/descompresión de video y la normativa existente en cuanto a los formatos de los archivos. También se explica todo lo necesario para crear y publicar archivos de video en una página Web.

Formato de Video MPEG

MPEG obtiene su nombre del Grupo de Expertos para la Imagen en Movimiento (Motion Picture Experts Group). Un punto sobre el que reina confusión en lo que se refiere a MPEG es que mientras que MPEG se refiere a un codec, también se utiliza para referirse a un formato de archivo. Realmente existen dos normativas codec: MPEG-1 y MPEG-2. MPEG-1 es la normativa empleada en Internet, mientras que MPEG-2 es una normativa de calidad profesional, empleada en radiodifusión.

La resolución de los videos MPEG-1 es de 352 x 240 píxeles, con una velocidad de 30 cuadros/segundo. Para ver un video MPEG a 30 cuadros/segundo hay que utilizar una tarjeta de video que incorpore un chip decodificador MPEG. Estos decodificadores se están incorporando cada vez más a todas las tarjetas de video. Los archivos de video MPEG

suelen tener una extensión .mpg. MPEG utiliza una característica de compresión llamada *cálculo predictivo*, un método que utiliza el cuadro de video activo para predecir qué es lo que van a contener lo cuadros siguientes. Este método de compresión hace que resulte imposible efectuar ediciones en los videos MPEG.

MPEG comprime y descomprime las imágenes y los sonidos a la misma velocidad. Precisamente es la elevada velocidad y la supercompresión lo que ha hecho de MPEG el formato estándar de video en Internet. MPEG es capaz de tratar datos descomprimidos de 1,2 a 1,5 Mbytes por segundo, y comprimirlos con una relación de 50:1, o más, antes de que se empiece a notar una degradación en la reproducción de un video.
También son posibles unas relaciones de compresión de 200:1, aunque con esa relación las imágenes se degradarán, a menos que se emplee un hardware de alta calidad. MPEG dispone del mayor nivel de compresión y genera la máxima calidad de video cuando se descomprime. Con MPEG suele utilizarse una compresión basada en hardware. La mayor desventaja es que el hardware de compresión tiene un costo alto en precio.

Formato AVI

El formato AVI (Audio/Video Interleaved, es decir Audio/Video Entrelazado) es un formato de Microsoft para video y sonido. Tal como indica el nombre del formato, los datos de video están entrelazados con los de sonido, dentro del mismo archivo, de modo que la porción de sonido de la película está sincronizada con la porción de video. AVI utiliza los codecs Indeo de Intel y Cinepak, que últimamente están muy de moda. Los archivos AVI se reproducen normalmente a unos 15 cuadros/segundo, en una ventana pequeña (de 320 x 240 píxeles). Con hardware o software de aceleración se pueden ejecutar secuencias de video AVI a 30 cuadros/segundo en una ventana grande o bien a pantalla completa. El formato AVI accede a los datos desde el disco duro sin necesidad de utilizar gran cantidad de memoria. La carga y reproducción es rápida, porque sólo se accede en cada momento a unos cuantos cuadros de video y a una breve porción de sonido. Los archivos AVI también se comprimen para aumentar la calidad de las secuencias de video y reducir su tamaño.

El formato AVI es el formato de video estándar par los archivos de video en Windows. Sin embargo, es raro encontrar archivos AVI en Internet. Esta situación probablemente irá cambiando según vaya variando la base de los usuarios que acceden a Internet, reflejando el gran número de usuarios de Windows que acceden a Internet.

Formato MPEG en Windows

Si se tiene la intención de incorporar videoclips en formato MPEG en las páginas Web, debe saber que para producir un clip de video MPEG de alta calidad se necesitará un codificador hardware que cuesta aproximadamente 4.000 dólares. De momento sólo hay una tarjeta codificadora MPEG de este precio, llamada Producer, de la empresa Sigma Designs, que también creó la placa Real Magic (que fue la primera tarjeta decodificadora MPEG que podía reproducir a pantalla completa videos en formato MPEG). Sigma fue una de las primeras compañías en ofrecer una tarjeta de video, llamada Rave, que incluía un chip decodificador MPEG.

4.5. Figuras Animadas

Para poder apreciar la construcción de una animación utilizando lo antes citado, en este proyecto se modelan las figuras de simbología marcada en la Guía Roji® y los centros de recopilación de los parámetros del agua en el Municipio de Cd. Nezahualcóyotl. El código fuente de la construcción de estas figuras ya se menciono en el capítulo II de este trabajo.

Los modelos se muestran en archivos con extensión **.AVI que pueden ser vistos en cualquier reproductor de Windows sin ningún problema, utilizando los programas Pov-Ray y VFD.

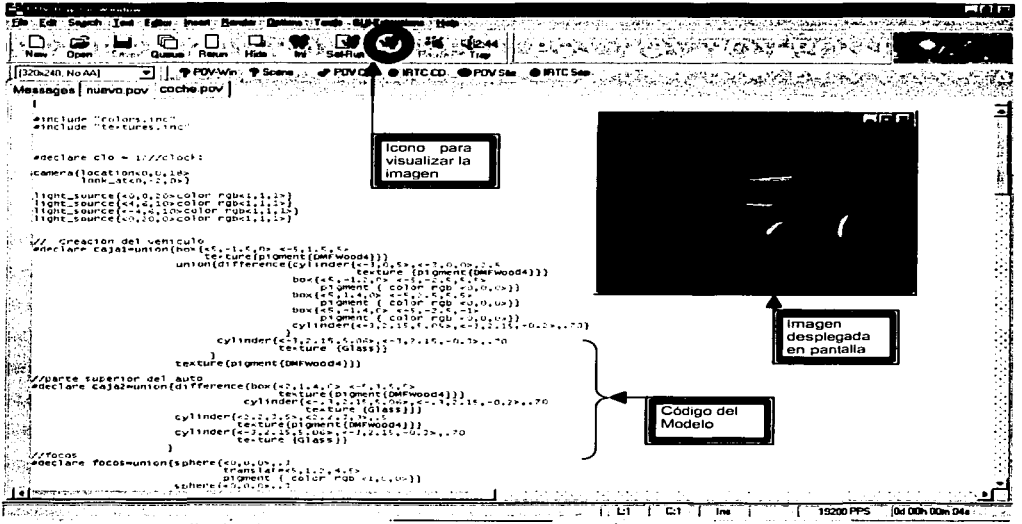


Fig. 4.1 Pantalla donde se muestran algunos elementos del área de trabajo del programa Pov-Ray

4.5.1. Secuencias de Archivos BMP

Como ya hemos visto una animación es la generación de secuencias de imágenes individuales llamadas cuadros que al unirse varios por un intervalo de tiempo construyen la animación deseada.

Para este proyecto utilizamos el lenguaje de programación llamado Pov-Ray versión 3.1a que como se vio en el capítulo 1, es un programa de generación de imágenes foto realistas de manera sencilla y de gran calidad.

Trabaja sobre la plataforma de Windows 9X, con una gran diversidad de opciones en pantalla para una mayor facilidad en su manejo, despliega la imagen en una pequeña ventana. Este lenguaje por default nos proporciona archivos de imagen con extensiones **.BMP que son los que utilizaremos para formar las secuencias de archivos de imágenes, para esto Pov-Ray nos brinda la facilidad de crear un nuevo archivo con extensión **.INI que tiene características específicas.

```
// ************************************
//            POZO
// ************************************

#include "colors.inc"
#include "textures.inc"

 #declare clo = 11//clock;

    camera{location<-2,6,25>
      look_at<-2,4,5>}

light_source{<0,1,20>color rgb<1,1,1>}
light_source{<7,5,10>color rgb<1,1,1>}
light_source{<-7,5,10>color rgb<1,1,1>}
//light_source{<0,10,0>color rgb<1,1,1>}

#declare llave=union{union{cylinder{<0,7.5,0>,<0,9,0>,0.8}
                          sphere{<0,9,0>,0.8}
                          cylinder{<2,9,0>,<0,9,0>,0.8}
                          sphere{<2.7,9,0>,1.2}
                          cylinder{<3.5,9,0>,<5,9,0>,0.8}
                          cylinder{<2.7,10,0>,<2.7,11,0>,0.3}
                          texture{Gold_Texture}}
                    union{cylinder{<1.7,11.2,0>,<3.7,11.2,0>,0.4}
                          sphere{<1.5,11.2,0>,0.6}
                          sphere{<3.9,11.2,0>,0.6}
                          pigment{Red}}
                   }
```

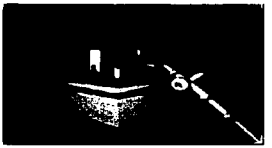


Fig.4.2 Generación de una imagen Pozo.bmp utilizando Pov-Ray.

4.5.2. Archivos INI

Son archivos de texto que nos sirven de contadores incrementándose de uno en uno los valores para generar las secuencias de imágenes BMP, además, cuenta con parámetros de calidad, resolución, etc., estos archivos son leídos y compilados por el mismo programa de Pov-Ray como archivos individuales, pero que están ligados a un archivo **.POV que contiene la escena que formara la animación.

Para esto es necesario que en el archivo **.POV se declare una variable con la letra K, que contendrá y ligará la información del archivo *.INI con el archivo **.POV.

La variable K dentro del archivo **.POV puede ser utilizada en cualquiera de las directivas e instrucciones que utiliza Pov-Ray con el fin de automatizar la generación de las imágenes, es decir:

En esta directiva tenemos una caja fija de coordenadas 1,1,1 hasta -1,-1,-1

```
Box {<1,1,1> <-1,-1,-1> }
```

Si declaramos en las coordenadas ya sea en una en particular o en todas la variable K.

```
Box {<K,1,1> <-1,K,-1> }
```

Harán que la caja sea dinámica dependiendo como configuremos el archivo **.INI, ya que podemos hacer que cambie de forma, tamaño, lugar, etc., en cada imagen individual.

Nuestra variable K se puede acompañar de las operaciones básicas de suma, resta, multiplicación, división con el objetivo de poder hacer movimientos más lentos o rápidos en nuestra animación.

Esto es por que en el Archivo **.INI declarado tiene la función principal de indicar el número de cuadros en que se inicia nuestra secuencia hasta el cuadro final al generarse uno a uno, regresa este valor del contador obtenido, lo guarda en la variable K, lo manda al archivo **.POV que toma este nuevo valor para cada una de las directivas que está ligada a esta variable, dando el efecto de movimiento al unir la secuencia.

La figura siguiente nos muestra una pantalla del programa de Pov-Ray leyendo un archivo POZO.INI y sus directivas de control que nos permitirá crear la secuencia de archivos **.BMP para posteriormente generar la animación.

```
Input_File_Name=pozo.POV

Initial_Frame=00
Final_Frame=20

Initial_Clock=00
Final_Clock=21

Cyclic_Animation=on
Pause_when_Done=off
```

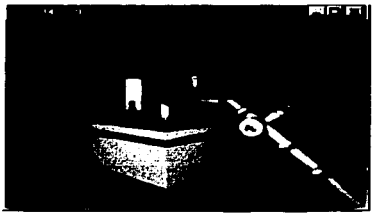


Fig. 4.3 Construcción del archivo .ini tomando como base el archivo Pozo.pov utilizando Pov-Ray.

Es necesario explicar estas directivas para un mejor control de la variable K que tiene el control y es enlace entre los Archivo INI y POV a continuación:

Input_File_Name=pozo.POV

Nos indica el nombre del archivo POV, dónde se encuentra la escena a generarse y debe de contener la variable K; en este caso, hace referencia al archivo Pozo.pov de nuestro proyecto. La variable K hace que el modelo en 3D haga un movimiento con respecto al eje Y para que la veamos girar 360 grados.

Initial_Frame=00

Final_Frame =20

Estas dos directivas nos indican la secuencia de inicio y final de los cuadros a generar; aquí nos indica que la secuencia de los cuadros será Pozo00.bmp hasta Pozo20.bmp y que se podrá ver que son 21 cuadros y no 20 contando la imagen 0.

Initial_Clock=00
Final_Clock=21

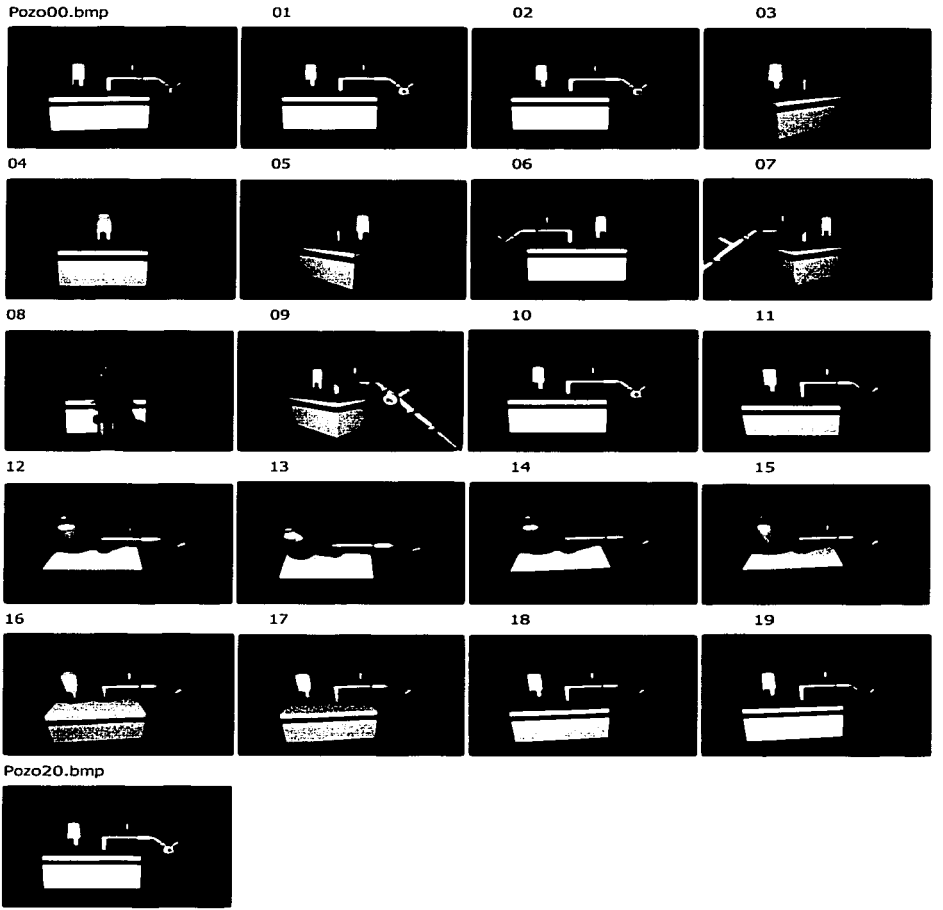
Esta dos directivas son muy importantes ya que forman el contador de la variable K, y nos darán el valor de cada imagen del archivo Pozo.pov hasta completar nuestra secuencia.

Cyclic_Animation=on
Pause_when_Done=off

Estas instrucciones nos indican que la animación a realizar es cíclica y que no para entre imagen e imagen.

A continuación se muestran dos secuencias una de 21 imágenes y otra de 11 imágenes renderizadas en el programa de Pov-Ray utilizando los Archivos Pozo.INI, Pozo.POV, Muneco.INI y Muneco.POV que son modelos en tercera dimensión de este proyecto.

Fig. 4.4 Secuencia de 21 Archivos de imagen: Pozo00.bmp → Pozo20.bmp



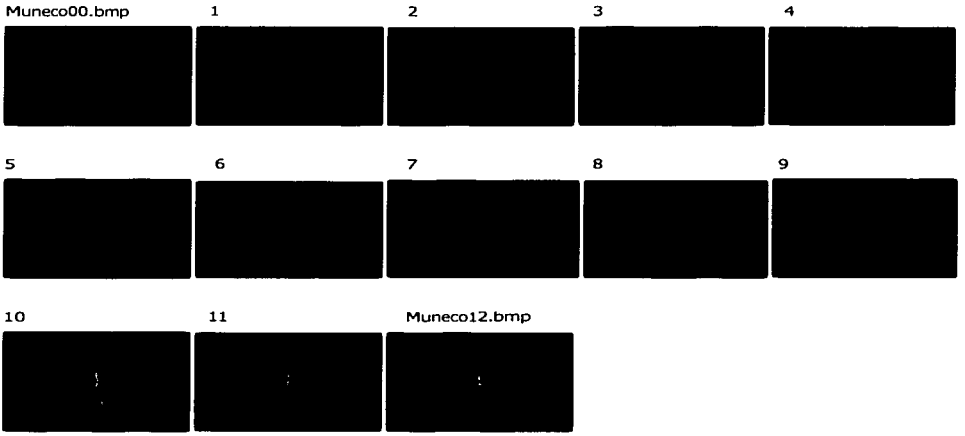
```
// ************************************
//            MUÑECO
// ************************************

#include "colors.inc"

#declare clo = clock;

camera{location<0,4,18>
       look_at<0,2,5>
      }
light_source{<0,1,15>color rgb<1,1,1>}
light_source{<7,5,10>color rgb<.51,.3,.51>}
light_source{<-7,5,10>color rgb<.51,.33,.51>}

#declare muneco=union{sphere{<0,4.5,0>0.5}
                      sphere{<0,5.5,0>0.8}
                      cylinder{<0,0,0>,<0,4,0>,0.8}
                      cylinder{<0,0,0>,<2,-4,0>,0.5}
                      cylinder{<0,0,0>,<-2,-4,0>,0.5}
                      cylinder{<-2,4,0>,<2,4,0>,0.4}
                      cylinder{<-2,4,0>,<-2,6,0>,0.3}
                      cylinder{<2,4,0>,<2,6,0>,0.3}
                      sphere{<2,4,0>0.3}
                      sphere{<-2,4,0>0.3}
                      pigment{Green}
                     }

object{muneco
       rotate y*clo*30}
```

Fig. 4.5 Código del archivo en muneco.pov

```
Input_File_Name=muneco.POV


Initial_Frame=00
Final_Frame=12
Initial_Clock=00
Final_Clock=13


Cyclic_Animation=on
Pause_when_Done=off
```

Fig. 4.6 Código del archivo en muneco.ini

Fig. 4.7 Secuencia de 13 Archivos de imagen: Muneco00.bmp → Muneco12.bmp



4.5.3. Construcción de archivos animados *.AVI utilizando el VFD

Después de declarar el tipo de imagen a utilizar y generar los cuadros necesarios de la secuencia procedemos a unir todas estas imágenes renderizadas para crear nuestra animación. Ésta la conjuntamos en un solo archivo con extensión **.AVI que es estándar y que puede ser reproducida casi por cualquier plataforma de cómputo.

Para hacer esta animación utilizaremos el programa VFD que ya analizamos en este mismo capítulo, observamos que este programa se obtienen la secuencia de imágenes, las procesa, comprime y genera el archivo **.AVI, tomando en consideración los parámetros que se indiquen.

Tomando las 21 imágenes generadas pozo??.bmp del archivo Pozo.pov mostramos a continuación el proceso de creación de la animación:

Primeramente en modo de Sistema Operativo MS-Dos

Se teclea en la línea de comando: C:\Tesis>VFD pozo??.bmp -A1 -S2

Esta línea le indica al programa que recolecte todos los archivos pozo??.bmp que forman la secuencia de la animación, con el parámetro (–A1) nos indica que se construirá como un archivo Pozo.avi por defaul, el parámetro (–S2) nos indica que son dos cuadros por segundo en la secuencia.

Se muestran 4 pantallas dónde se lleva a cabo el proceso de crear una animación al emplear el Programa VFD.

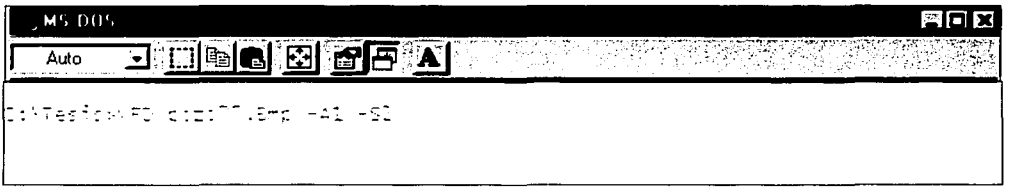


Fig.4.8 Se escribe en la línea de comando el archivo y los parámetros a emplear

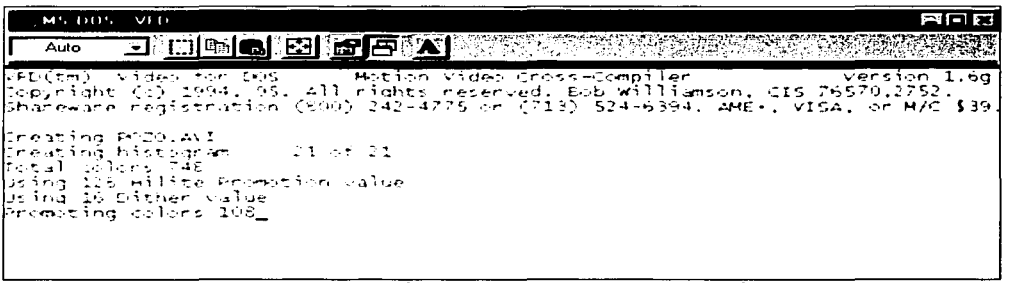


Fig. 4.9 Inicia un análisis de las estructuras de las imágenes y crea el archivo animado Pozo.avi

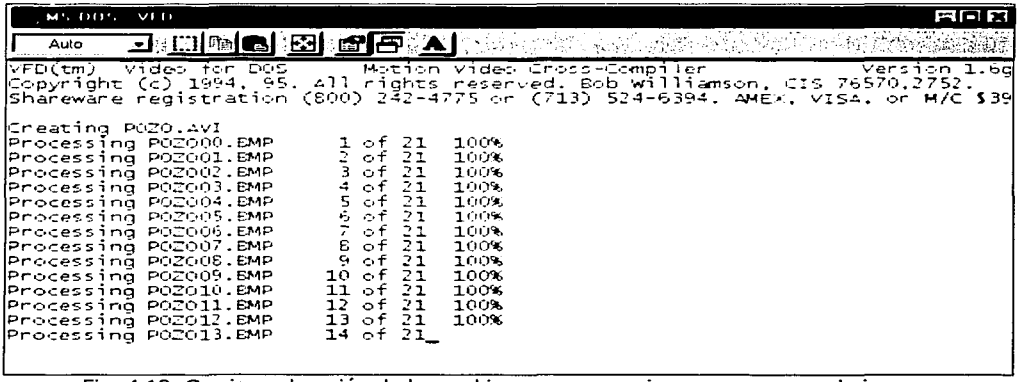


Fig. 4.10 Comienza la unión de los archivos en secuencia como vemos en la imagen

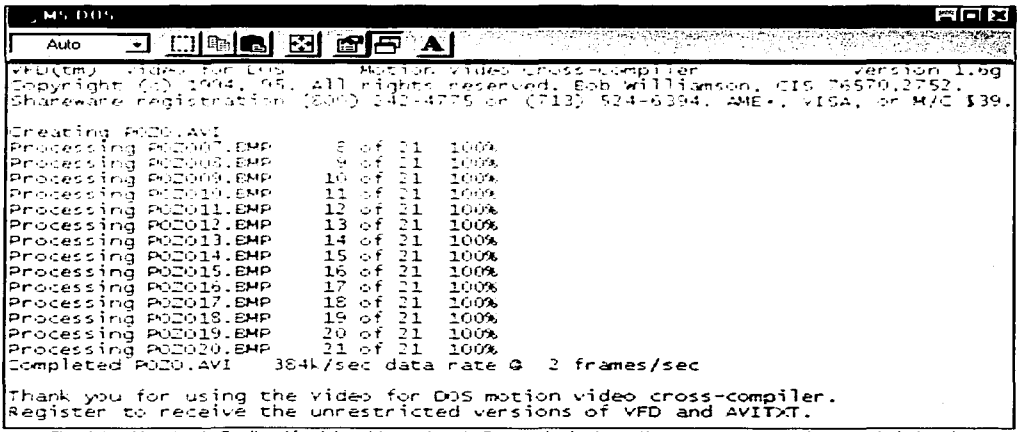


Fig. 4.11 Muestra la finalización del archivo animado Pozo.avi, al mismo tiempo nos muestra las propiedades de: tamaño y total de los cuadros por segundo a la que se generó.

Al finalizar este proceso, visualizamos como el archivo generado de la animación con el programa VFD se guarda en un archivo con el nombre Pozo.avi, esté lo podemos observar en cualquier reproductor de medios como el caso siguiente:

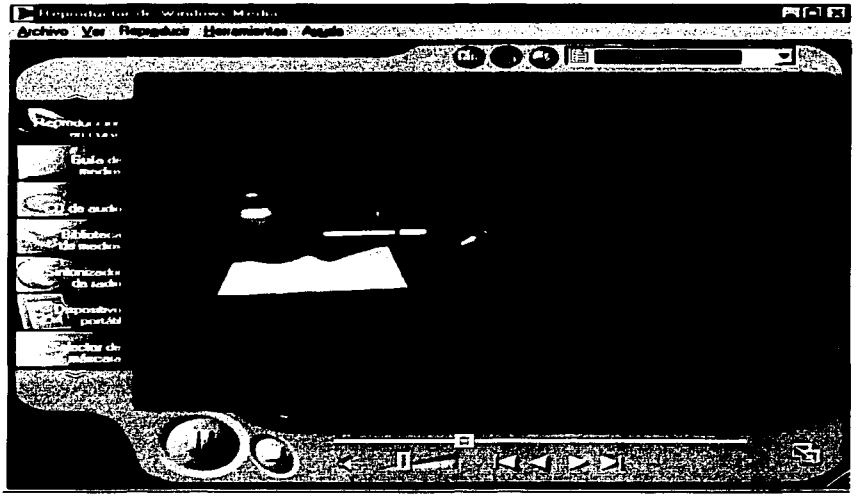


Fig. 4.12 Pantalla donde es mostrado el archivo animado Pozo.avi utilizando el Reproductor de Windows Media Player de Microsoft.

Todos los objetos que son utilizados en este proyecto fueron desarrollados de una manera similar.

4.5. Recorrido Animado

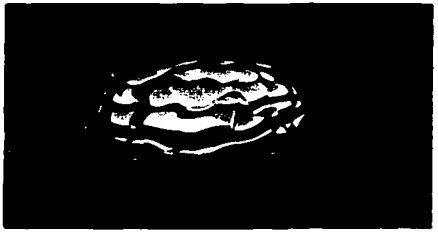
Para una mejor representación de la recopilación de los parámetros de contaminación del agua, se ha diseñado una animación representativa del recorrido que realizan los Ingenieros del Laboratorio de Ambiental del Centro Tecnológico Aragón por el Municipio de Cd. Nezahualcóyotl.

Para tal animación se dibujó todo el Municipio de Cd. Nezahualcóyotl y la parte Alta de Aragón, representándolos con rectángulos y otras figuras de color azul, también se colocaron los modelos tridimensionales de los centros de acopio de información y otras figuras representativas de la Guía Roji®. En un coche tridimensional se simula el recorrido por las calles y avenidas del Municipio que tienen como inicio el Campus Aragón.

La animación está compuesta por 400 cuadros o imágenes generados en el programa de Pov-Ray que posteriormente se unen con el programa VFD a 15 cuadros por segundo, para convertirla en un archivo de video con extensión AVI, este procedimiento es realizado de igual manera que las figuras y gráficas antes citadas.

CAPÍTULO 5

Resultados en Internet



El por qué de la publicación en el World Wide Web

Desde la invención de la imprenta, realizada por Johannes Gutenbeig hizó avanzar la economía y produjo cambios en el comercio, la política, la sociedad, la literatura y la ideología que marcaron el comienzo del Renacimiento. El World Wide Web está anunciando una nueva generación de publicaciones, llevándonos hacia el hipertexto, la multimedia y la red global. El Web está creciendo a una velocidad sorprendente y está modificando el mundo de las publicaciones, haciendo posible que cualquier persona pueda publicar información y ponerla al alcance de todas las personas en el mundo entero.

En el entorno actual de los negocios competitivos, global y rápidamente cambiantes, es crucial que la información actual esté inmediatamente disponible para el consumidor que la necesita. El World Wide Web permite publicar rápidamente tanto información de mercado, como de servicio al cliente o de investigación, a través de una dirección única. El Web también es un gran foro para la expresión personal que permite compartir las ideas y los temas de interés con otras personas.

5.1. El World Wide Web

El proyecto del World Wide Web lo comenzó, en 1989, Tim Berners-Lee en el laboratorio de física de alta energía del CERN (Centro Europeo de Investigación Nuclear). El objetivo de este proyecto era encontrar un modo de compartir la investigación y las ideas con los demás empleados e investigadores del centro, repartidos por todo el mundo. En su propuesta inicial, el Web se llamó "proyecto de hipertexto". *Donde el Hipertexto* es un

término acuñado por Ted NelsCon en los años sesenta, que hace referencia a todo texto que contiene conexiones con otros documentos, de modo que el usuario puede hacer *clic* sobre una palabra o frase para obtener información adicional sobre temas relacionados. La *Hipermedia* es un término más amplio para los documentos que incluyen información en formato multimedia, tales como sonido y video; técnicamente el *World Wide Web* hace referencia al ciberespacio abstracto de información, mientras que *Internet* hace referencia a la parte física de la red, es decir, al hardware formado por los cables y los ordenadores.

Donde sus características son:

- Información por hipertexto: Diversos elementos (texto o imágenes) de la información que se nos muestra en la pantalla están vinculados con otras informaciones que pueden ser de otras fuentes. Para mostrar en pantalla esta otra información bastará con hacer *clic* sobre ellos.

- Gráfico: En la pantalla aparece simultáneamente texto, imágenes e incluso sonidos.

- Global: Se puede acceder a él desde cualquier tipo de plataforma, usando cualquier navegador y desde cualquier parte del mundo.

- Pública: Toda su información está distribuida en miles de ordenadores que ofrecen su espacio para almacenarla. Toda esta información es pública y toda puede ser obtenida por el usuario.

- Dinámica: La información, aunque esta almacenada, puede ser actualizada por el que la publico sin que el usuario deba actualizar su soporte técnico.

- Independiente: Dada la inmensa cantidad de fuentes, es independiente y libre.

5.1.1. Cómo Funciona una Publicación en el Web

La publicación en el Web se efectúa mediante el modelo de cliente-servidor. Un *servidor Web* es un programa que funciona en un ordenador configurado para facilitar documentos a otros ordenadores que efectúan peticiones sobre esos documentos. Un *cliente Web* es un programa que permite que un usuario solicite documentos a un servidor. Como el servidor sólo trabaja cuando se solicita el documento, el método utilizado es eficaz para compartir documentos, porque el cliente sólo necesita una pequeña parte de los recursos del servidor.

Un servidor tiene la función de proporcionar al usuario la información requerida por medio de un navegador que es el programa que nos va a ofrecer el acceso a Internet, este contiene las herramientas necesarias para comprender e interpretar el lenguaje utilizado por el servidor Web.

Este lenguaje es el HTML que sus siglas corresponden con la definición "Lenguaje para Marcado de Hipertexto" tratándose de un lenguaje para estructurar documentos a partir de texto en World Wide Web. Este lenguaje se basa en etiquetas (instrucciones que le dicen al texto, objetos, imágenes, etc., como deben mostrarse) y atributos (que son parámetros que le dan un valor a la etiqueta).

El protocolo que utilizan los servidores y los clientes Web para comunicarse entre ellos se llama HTTP (HyperText Transmission Protocol, es decir, Protocolo de Transmisión de Hipertexto) que consiste en la facilidad para la transferencia de información, para así mandar los documentos de tipo HTML a través de Internet y así poder atender las solicitudes de archivos por parte del navegador. Todos los servidores y los clientes Web deben ser capaces de comunicarse por este protocolo HTTP para enviar y recibir sin ningún problema los documentos hipermedia.

5.1.2. El Navegador y el URL de la Web

Un cliente Web, usualmente llamado *navegador Web,* se conecta a la red mediante un ordenador al que se le ha asignado una dirección de la red, extrañamente conocida por URL (Uniform Resource Loeator, es decir, Localizador Uniforme de Recursos). El usuario envía una petición a un servidor Web en la que solicita un documento existente en el Web. El servidor responde enviando el texto y los demás elementos a los que haga referencia el hiperenlaces de los textos con imágenes, sonidos o videos.

Los documentos HTML, también llamados documentos Web, permiten que los usuarios hagan clic en una palabra o frase de hipertexto para acceder a archivos o saltar a otros documentos HTML. Estos enlaces de hipertexto entre archivos y documentos efectuados entre los servidores de todo el mundo hacen que el sistema funcione como si fuera una enorme telaraña (precisamente es lo que significa *web* en inglés) de información.

En los navegadores se permite que el usuario especifique una dirección del Web, llamada URL (Uniform Resource Locator, es decir, Localizador Uniforme de Recursos), y se conecte a un documento o recurso. Cuando se selecciona un elemento de hipertexto en un documento Web, el usuario realmente lo que está haciendo es enviar una petición para abrir un URL. Con un URL es posible representar prácticamente cualquier archivo o servicio de Internet. Un navegador Web puede actuar también como cliente FTP, Gopher y Telnet para proporcionar servicios y no solo consultas.

5.1.3. El Lenguaje Java

El más notable exponente de esta automatización del Web es Sun Computer, que ha lanzado un nuevo navegador Web, llamado HotJava, es un nuevo lenguaje interprete que se le llama Java. El cual nos permite un manejo más eficiente y proporcionar al mismo tiempo espacio para animaciones más interactivas entre los programadores y los usuarios,

el navegador carga el código necesario, llamado *applet,* y lo ejecuta. El código debe estar escrito en el lenguaje Java para su ejecución. De esta manera, la transacción completa consume muy poco tiempo (para su carga), pero puede ejecutarse durante más tiempo por sí misma.

Este lenguaje nos brinda la oportunidad y facilidad de poder hacer más atractivas nuestras publicaciones en el Web.

5.2. Organización de una Página Web

Para hacer una buena presentación Web lo ideal es crearnos un boceto inicial de la estructura. Si hacemos esto, no solo estamos procurando una presentación agradable y facilitando la tarea de navegar sino que también nos facilitamos el mantenimiento de futuras revisiones y modificaciones.

Debemos fijarnos los objetivos que queremos alcanzar según la información que vayamos a aportar. Para crear nuestra primera página, estos objetivos deberían no ser muy pretenciosos o tener un sentido únicamente personal. Tener claros los objetivos nos ayudara a no plasmar contenidos confusos o innecesarios.

Después de marcar los objetivos, hay que organizar el contenido por temas o secciones, que se ajusten a lo que necesitamos. Es conveniente que los temas sean razonablemente cortos y si fuera necesario dividir en subtemas. Si por el contrario tenemos temas muy cortos, lo correcto será agruparlos bajo un encabezado de tema algo más general.

El Primer Paso

Consiste de una o más páginas Web que contienen texto y gráficos y que están vinculadas entre si creando un cuerpo de información. La página principal o página base es desde donde se comienza a visitar la presentación y su URL será la que figure como dirección de

la presentación. Esta página base debe ofrecer un panorama general del contenido de la presentación.

La Organización
Es muy importante ya que consiste en la estructuración de la información recopilada en un conjunto de páginas Web. Podemos crearnos una estructura propia pero lo más lógico es guiarnos por una estructura clásica.

5.3. Estructura de una Página

Para publicar nuestros documentos en una pagina Web utilizamos el lenguaje HTML que nos servirá para estructurar de una manera muy sencilla y fácil nuestras paginas, ya que este lenguaje esta basado en etiquetas. La mayoría de los documentos tienen una estructura común como son: títulos, párrafos, listas, etc., que van a ser definidas por este lenguaje mediante etiquetas. Cualquier cosa que no sea una etiqueta es parte del documento mismo.

Este lenguaje no describe la apariencia del diseño de un documento sino que ofrece a cada plataforma que le de formato según su capacidad y la de su navegador como tamaño de la pantalla, fuentes que tiene instaladas, etc.

Básicamente, los documentos escritos en HTML constan del texto mismo del documento y las etiquetas que pueden llevar atributos. Esto llevado a la práctica, vendría a ser:

<etiqueta> texto afectado < / etiqueta>

La etiqueta del principio activa la orden y la última (que será la del principio precedida del signo /) la desactiva. No todas las etiquetas tienen principio y final pero esto lo veremos más adelante.

5.3.1. El Inicio de un Documento HTML

La estructura básica de un documento HTML: Cabecera y cuerpo del documento

Tres son las etiquetas que describen la estructura general de un documento y dan una información sencilla sobre él. Estas etiquetas no afectan a la apariencia del documento y solo interpretan y filtran los archivos HTML.

<HTML>: Limitan el documento e indica que se encuentra escrito en este lenguaje.

<HEAD>: Especifica el prólogo del resto del archivo. Son pocas las etiquetas que van dentro de ella, destacando la del título <TITLE> que será utilizado por los marcadores del navegador e identificará el contenido de la página. Solo puede haber un título por documento, preferiblemente corto aunque significativo, y no caben otras etiquetas dentro de él. En head no hay que colocar nada del texto del documento.

<BODY>: Encierra el resto del documento, el contenido.

```
<HTML>
<HEAD>
<TITLE>Ejemplo A</TITLE>
</HEAD>
<BODY>

   U.N.A.M.
   CAMPUS ARAGON

</BODY>
</HTML>
```

5.3.2. La Primera Vista

Las siguientes etiquetas no afectan a la apariencia del documento, solo interpretan y filtran los archivos HTML.

<H1>, <H2>, <H3>...: Titulares. Sirven para dividir el texto en secciones. Se pueden definir seis niveles de titulares, el texto que deseamos que sea un titular se pone entre las etiquetas <H1> Titular </H1>. Se definen mediante las etiquetas <H1>.....</H1> hasta <H6>.....</H6>

<P>: Párrafos. En principio, sin entrar en detalles de alineación u otras características, digamos que se definen por las etiquetas <P>.....<P>. Esta etiqueta, en un principio, se diseñó para saltar de párrafo por lo que puede ir sola "<P>" al final de un texto indicando que a continuación se quiere una línea en blanco aunque se recomienda abrirlas y cerrarlas.

: Saltos de línea. Esta etiqueta sirve para realizar un salto de línea, pueden ponerse tantas como se desee y realizará un salto de línea por cada una.

<!-- -->: Comentarios. Son directivas que nunca se mostrarán a través del navegador y que servirán para recordatorios en futuras revisiones del documento.

```
<HTML>
<HEAD>
<TITLE>Ejemplo B</TITLE>
</HEAD>
<BODY>
  <H1>Un Ejemplo</H1>
  <!-- Aqui va un comentario que no es
    interpretado por el navegador -->
<P>
   U.N.A.M.
   CAMPUS ARAGON
    Proyecto de Tesis y Tres Saltos de Linea.
</P>
 <br>
 <br>  < !-  Tres Saltos de linea ->
 <br>
</BODY>
</HTML>
```

5.3.3. Creación de Enlaces

Lo característico del lenguaje HTML es el poder generar vínculos de hipertexto para enlazar con ellos todos sus documentos en web.

Para generar un enlace a otro documento necesitamos el nombre de un archivo (o su dirección URL) y el texto que servirá de punto de activación del otro documento. Este segundo elemento será el que veamos en pantalla y que se servirá del primero para saltar de documento.

Los enlaces se generan mediante la etiqueta <A>.....</A> y, a diferencia de los vistos anteriormente, llevará siempre dentro de la etiqueta un atributo ya sea:

<A HREF=""> o <A NAME="">.

<A HREF="URL">.....</A>: Es el más habitual de los atributos y sirve para saltar entre diferentes URLs, mientras que <A HREF="archivo2.html">.....</A > nos manda a un archivo.

La instrucción <A NAME="parte1">Primera parte</A>: se utiliza el atributo name para dar nombre a una sección de nuestro documento y dirigirse a esa sección.

```
<HTML>
<HEAD>
<TITLE>Ejemplo C</TITLE>
</HEAD>
<BODY>
  <A NAME="arriba"><H1>Página de enlaces</H1></A>
  <A HREF="#abajo">Ir abajo</A><br>

  <A HREF="ej4.html">Ir a ejemplo 4</A><br><br>
  <A HREF="http://www.unam.MX.com/">Ir a Vista</A><br>

  <br>.<br>.<br>.<br>.<br>.<br>.<br>.<br><br>
  <br><br><br><br><br><br><br><br><br><br><br><br>
  <br><br><br><br><br><br><br><br><br><br><br><br>
  <br><br><br>.<br>.<br>.<br>.<br>.<br>

  <A NAME="abajo"><br></A>
  <A HREF="#arriba">Ir arriba</A>
</BODY>
</HTML>
```

5.3.4. Listas

Junto con encabezados y párrafos, son otro de los elementos HTML más comunes. Pueden darse tres tipos diferentes de listas.

Listas numeradas u ordenadas: Se engloban por las etiquetas <ol>..... y cada elemento de la lista estará encabezado por la etiqueta <li> que puede o no llevar la etiqueta de cierre .

Listas con viñetas o sin orden: Se engloban por las etiquetas <ul>..... y cada elemento de la lista, también estará encabezado por la etiqueta <li>. El resultado es que el navegador inserta viñetas (marcadores) delante de cada elemento.

Listas de glosario: Cada elemento de la lista está compuesto por un término y una definición y cada una de estas partes tiene su propia etiqueta. Estas listas se engloban con las etiquetas <dl>.....</dl>. Para el término se usa la etiqueta <dt> y para la definición la etiqueta <dd>.

5.3.5. Estilos de Carácter

Estos estilos son etiquetas que afectan a palabras o caracteres dentro de otras entidades de HTML modificando el aspecto de ese texto para que sea diferente del texto que lo rodea. Existen dos tipos de estilos:

Estilos lógicos

Indican como se va a emplear el texto que realzamos, no el como se va a formatear.

.....:	Indica que los carácteres estarán enfatizados de alguna manera, generalmente en cursiva aunque dependerá del navegador.
.....:	Los carácteres tendrán mayor énfasis, generalmente en negrita.
<code>.....</code>:	Muestra como una fuente monoespaciada, generalmente Courier.
<samp>.....</samp>:	Muy similar a code.
<kdb>.....</kdb>:	Texto que el usuario debe escribir.
<var>.....</var>:	Nombre de una variable que deba ser reemplazada por su valor real. Generalmente en cursiva o subrayada.
<dfn>.....</dfn>:	Se usa para resaltar una palabra que se va a definir.
<cite>.....</cite>:	Se usa para citas cortas.

Estilos físicos

Modifican la presentación real del texto.

.....:	Pone el texto en negrita.
.....:	Pone el texto en cursiva.
<tt>.....</tt>:	Pone el texto en fuente monoespaciada.
<u>.....</u>:	Subraya el texto afectado.

5.3.6. Saltos y Líneas

Líneas divisorias: Se crean con la etiqueta <hr> que no tiene etiqueta de cierre ni lleva texto asociado. Se puede especificar el ancho de la línea con el siguiente atributo: <hr width="80"% >

Saltos de línea: La etiqueta
 inserta un salto de línea donde se coloque. Puede colocar tantas como desee y se insertará un salto de línea por cada una de ellas.

5.3.7. Tablas

Las tablas nos permiten representar y ordenar cualquier elemento de nuestra presentación en diferentes filas y columnas de modo que podamos resumir grandes cantidades de información de una manera que puede representarse rápida y fácilmente. El contenido de una tabla lo debemos desarrollar entre las etiquetas <table>.....</table>.

Cada fila de la tabla se indica mediante las etiquetas <tr>.....</tr>. Las etiquetas <th> y <td> con sus correspondientes etiquetas de cierre nos indican las filas individuales dentro de cada fila. Las etiquetas <th>.....</th> indican que se trata de celdas que sirven como encabezado de tabla y suelen visualizarse en negrita. Las etiquetas <td>.....</td> indican que se trata de celdas comunes.

```
<HTML>
<HEAD>
<TITLE>Ejemplo D</TITLE>
</HEAD>
<BODY>

<H1>Tablas de Ejemplo</H1>

<TABLE BORDER="1">
<TR>
<TH>Cabereca 1</TH>
</TR>
<TR>
<TD>Dato 1</TD>
</TR>
<TR>
<TD>Dato 4</TD>
</TR>
</TABLE>

</BODY>
</HTML>
```

Una vez colocadas las celdas, hay que alinear los datos dentro de cada celda. Así, dentro de cada etiqueta de celda podemos encontrar los atributos:

align= Define horizontalmente los datos al margen izquierdo (left), al derecho (right) o centrado (center).

valign= Define verticalmente los datos en la parte superior (top), en la parte inferior (bottom) o centrado (middle).

Podemos modificar el aspecto de la tabla cambiando el ancho de los bordes, el espaciado entre celdas y el ancho de las mismas.

width= Acompaña a <table> y especifica el ancho de la tabla, tanto en número de píxeles como en porcentaje respecto al ancho de la pantalla. También puede acompañar a las etiquetas <th> o <td> para especificar el ancho de las columnas.

Border= Anteriormente, ya hemos hablado de este atributo. Ahora le diremos que puede darle un valor que indicará el ancho del borde en píxeles. Border="0" indicaría la ausencia de borde.

Cellspacing= Suele acompañar a la etiqueta <table>. Indica el número de píxeles que separan cada celda. El valor predeterminado suele ser 2.

Cellpadding= También acompaña a la etiqueta <table>. Indica el espacio en píxeles entre el borde de la celda y su contenido. El valor predeterminado suele ser 1.

5.3.8. Imágenes

El uso de imágenes es uno de los factores que ha popularizado tanto World Wide Web, es incluir imágenes en una presentación Web que es muy sencillo, solo debe de tener en cuenta que las imágenes tienen que tener los formatos GIF, JPEG o PNG. Las imágenes en línea, se especifican a partir de la etiqueta <img> que no tiene una etiqueta correspondiente de cierre pero que puede acompañarse de los siguientes atributos:

src= Este atributo es obligatorio e indica el nombre del archivo de imagen (entre comillas) o la URL que se va a representar.

Align= Permite controlar la alineación de una imagen con respecto a una línea de texto adyacente o a otras imágenes en esa línea. Los tres valores posibles son los ya conocidos left, right, top, middle y bottom.

Alt= Es la alternativa que se estableció cuando todavía existían visualizadores de solo texto. Entre comillas podremos escribir un texto que suplantara a esta imagen si no se carga o mientras se carga, para visualizar la imagen, pasamos el ratón por encima.

WIDTH= Este atributo es opcional pero es recomendable ponerlo para ayudar al navegador a representar la imagen, significa el ancho de la imagen que vamos a representar.

HEIGHT= Al igual que el atributo WIDTH, es opcional y recomendable ponerlo, este significa el alto de la imagen.

BORDER= Con BORDER especificamos el ancho de un borde que rodea la imagen.

En la siguiente etiqueta se solicita y muestra una imagen con propiedades específicas como son: 180 píxeles de ancho, 120 de largo, sin borde, y un mensaje de pre-visualización.

```
<IMG SRC="/graficos/bebe1.jpg" WIDTH=140 HEIGHT=210 BORDER=0 ALT="Un bebé">
```

5.4. Creación de la Página de Resultados

5.4.1. Estructuras y Enlaces

A continuación se describe de forma general la estructura de la página de Internet así como sus ligas o enlaces que existen entre ellas:

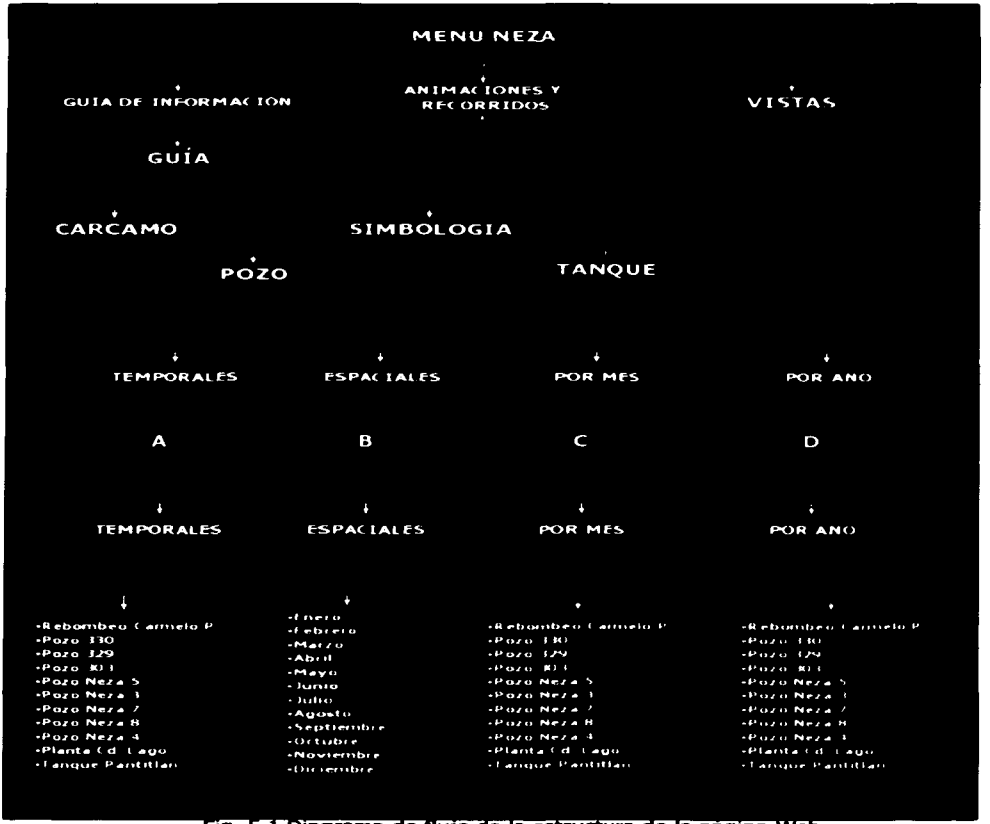


Fig. 5.1 Diagrama de flujo de la estructura de la página Web

5.4.2. Página Web Terminada

El resultado de la creación de la página Web se muestra en las siguientes pantallas:

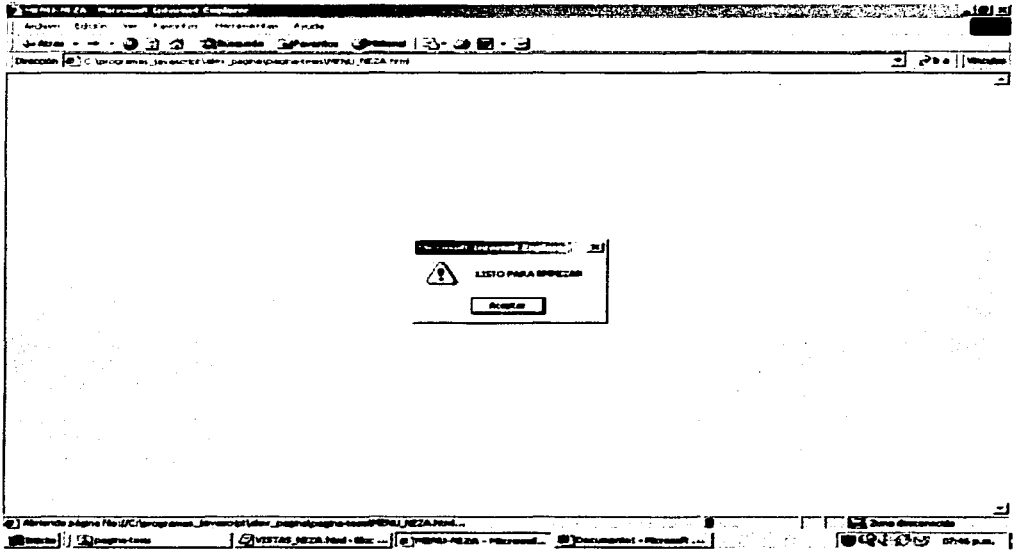


Fig. 5.2 Ventana de bienvenida al iniciar el proceso de resultados

Página Menú

En el menú principal encontramos los datos de la escuela y personales que hacen referencia a este proyecto y a la página Web, en la parte de abajo aparecen los enlaces que redireccionan hacia las ligas de las diversas gráficas y animaciones que se han tratado en los capítulos anteriores. También se localiza un icono en forma de puerta que será utilizada para poner información adicional sobre el tema en cualquier momento.

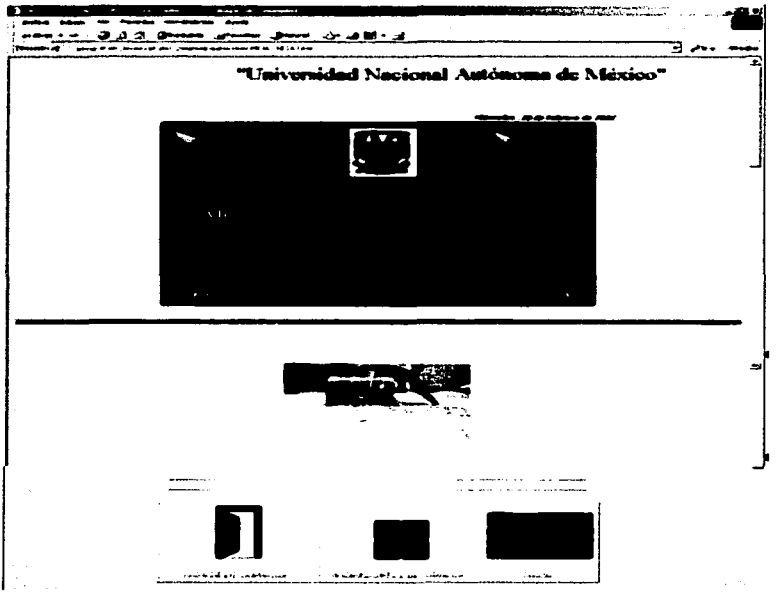


Fig.5.3 Menú principal

Página de Animaciones

En esta sección veremos las animaciones del recorrido realizado por el CTA en el Municipio de Nezahualcóyotl pasando por cada uno de los pozos, tanques y cárcamos, al igual que las animaciones de los objetos representativos.

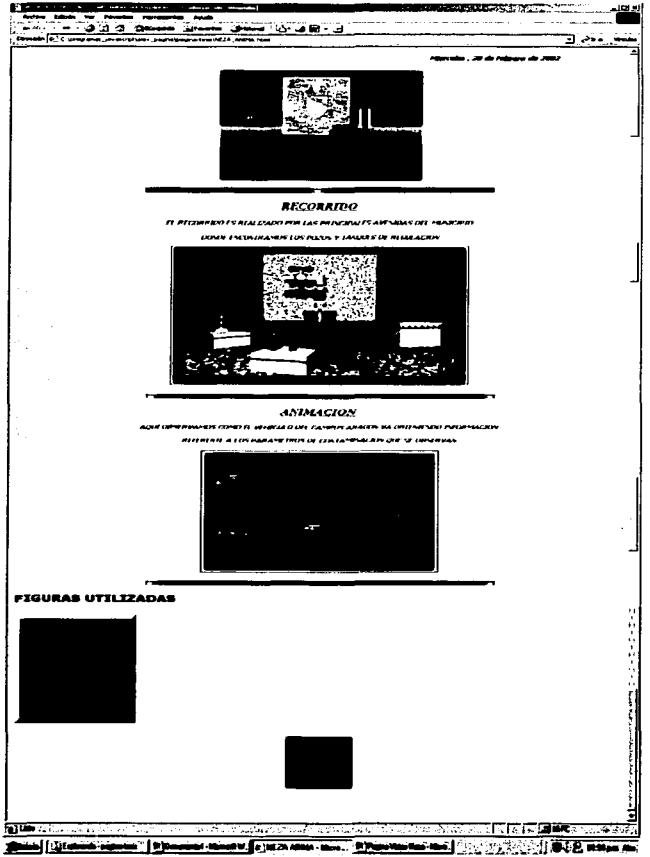


Fig. 5.4 Animación del recorrido del CTA

Visualizaciones Gráficas

En esta parte se puede visualizar las animaciones temporales que es el pozo contra los meses del año, pasando uno a uno mostrando sus parámetros de contaminación en dicho mes. Espaciales mes contra pozo que visualiza mes con mes pasando todos los pozos y mostrando los niveles de contaminación de cada uno de ellos; por otro lado en la parte de parámetros por mes se ven las vistas fijas de cada pozo, cárcamo y tanque contra los parámetros de contaminación, los parámetros por año muestran el comportamiento de un parámetro específico de contaminación en el transcurso del año.

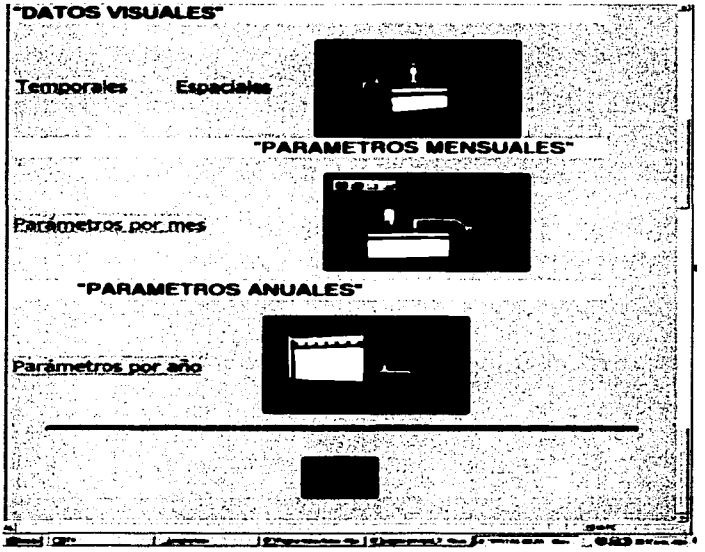


Fig. 5.5 Ventana de parámetros

Fig. 5.6 Submenú de la ventana de parámetros

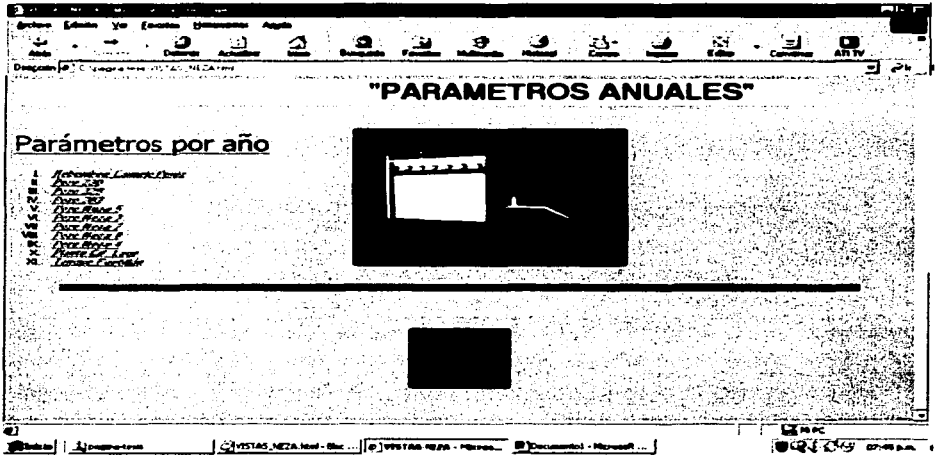


Fig. 5.7 Parámetros Anuales

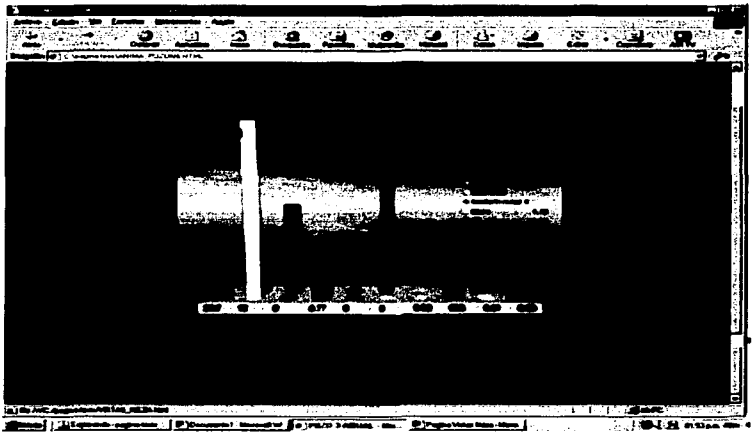


Fig. 5.8 Animación Temporal

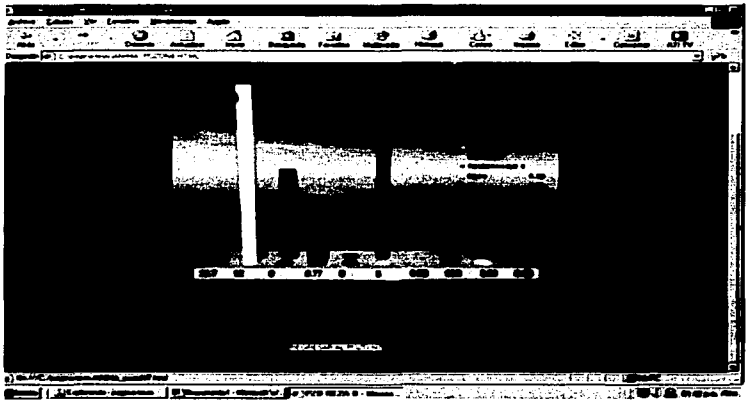


Fig. 5.9 Animación Temporal

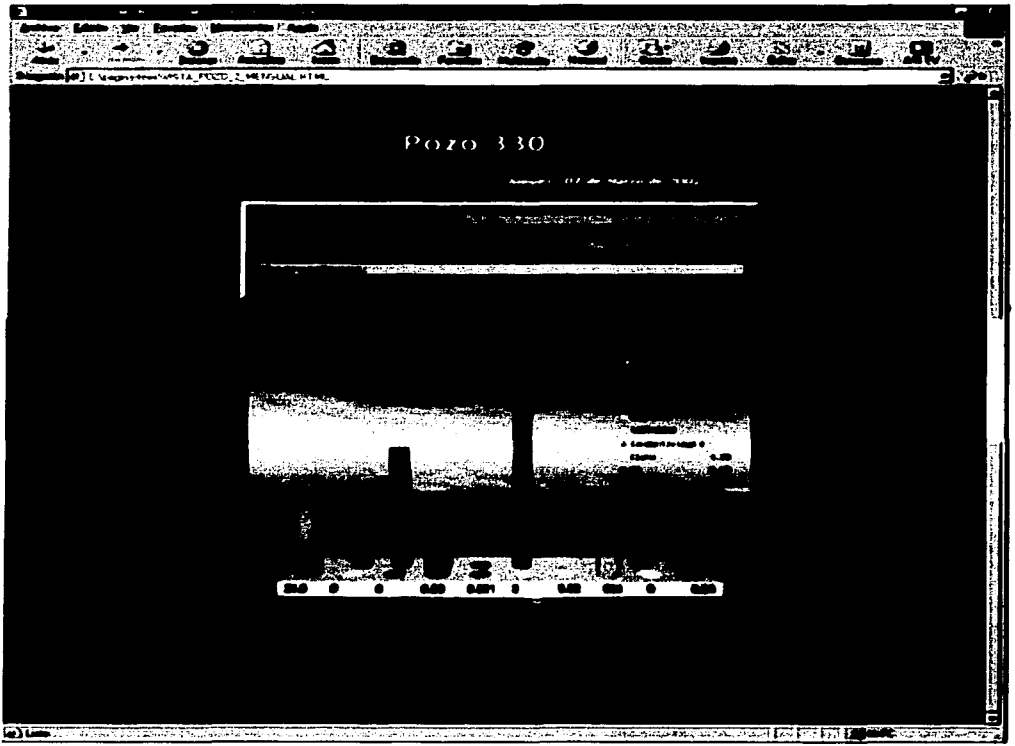


Fig. 5.10 Vista fija de los parámetros de contaminación por mes

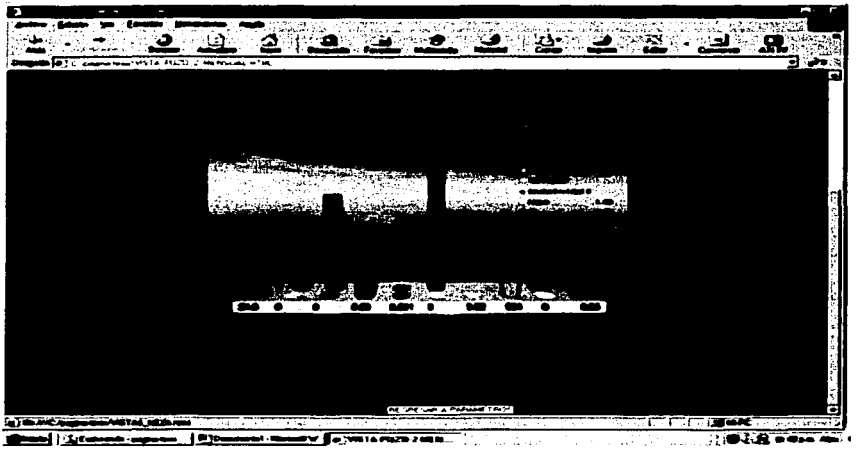


Fig. 5.11 Animación espacial

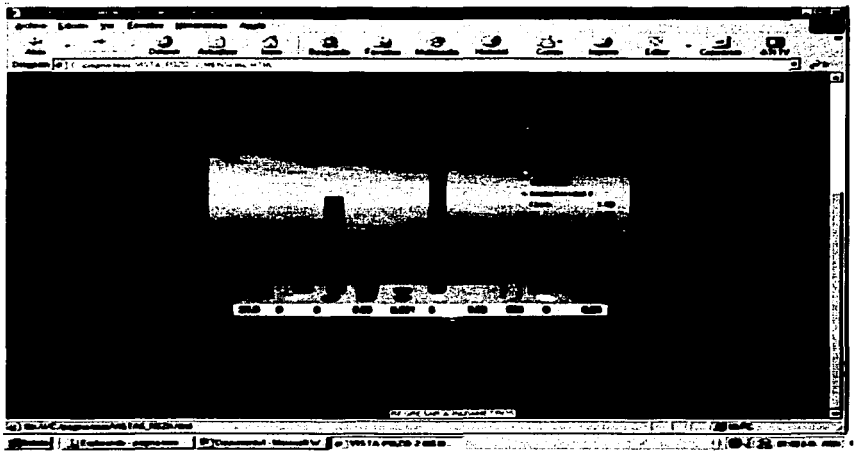


Fig. 5.12 Animación espacial

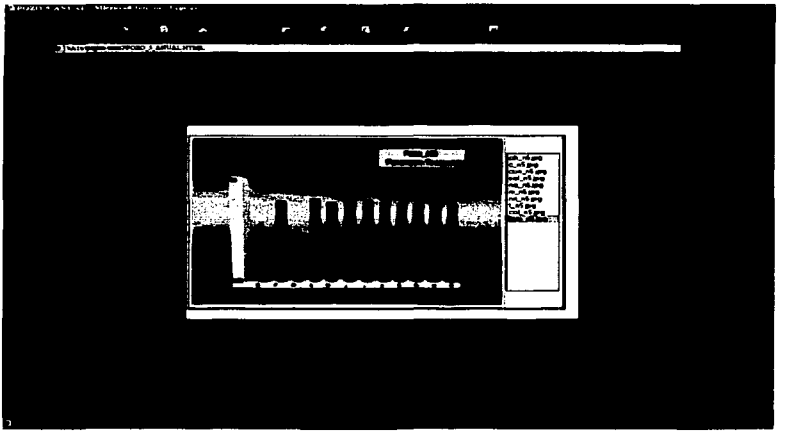


Fig. 5.13 Gráficas fijas anuales por parámetro

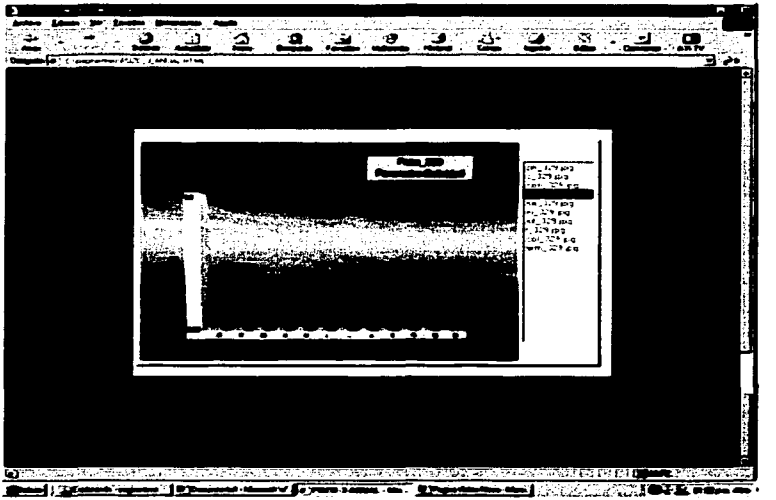


Fig. 5.14 Gráficas fijas anuales por parámetro

Conclusiones

La visualización gráfica por computadora inicia como una técnica para resaltar la información desplegada y generada por una computadora. Esta habilidad para interpretar y representar datos numéricos en imágenes tiene significativo incremento en la habilidad de las computadoras para presentar la información al usuario en una forma clara y entendible. Grandes cantidades de datos son rápidamente convertidos en gráficas de barra, gráficas redondas e imágenes. Los despliegues gráficos tienen posibilidades para auxiliarnos a entender sistemas complejos.

En tiempo pasado fueron muchas las razones por las que los medios de despliegue gráfico no fueron utilizadas. Primero, el ambiente de tiempo compartido en los 70's no era disponible para las gráficas. Los requerimientos de procesamiento gráfico simultáneo de varios usuarios trae consigo ineficiencia en el tiempo de respuesta de los otros usuarios del sistema. Una segunda razón de la carencia de gráficos son los costos de estos sistemas de despliegue, ya que son muy elevados.

En los 80's, sin embargo, con la introducción de microcomputadoras y pequeñas estaciones de trabajo, la complejidad en instalaciones de cómputo cambió. Se liberaron los precios de microprocesadores y memoria.

Aparecen entonces las computadoras personales con pantallas gráficas de barrido, como Apple Machintosh, IBM y su familia que popularizan los gráficos de mapas de bits para la interacción con el usuario.

Poco a poco las computadoras se han convertido en una herramienta poderosa de hecho, no existe ninguna área en que no se puedan aplicar las gráficas por computadora con algún beneficio y como consecuencia, no es sorprendente encontrar que se haya generalizado tanto el uso de la visualización gráfica, a pesar de que las primeras aplicaciones en la ingeniería y la ciencia debían depender de equipo costoso y complicado,

los avances en la tecnología de la computación han hecho que las gráficas interactivas por computadora sean una herramienta práctica.

La visualización gráfica constituyó un pequeño y especializado campo hasta principios de la década de 1980, sobre todo debido al elevado costo del hardware y por consiguiente los escasos programas de aplicación.

Hoy en día la visualización gráfica por computadora es utilizada de manera rutinaria en diversas áreas, como en la ciencia, ingeniería, empresas, industria, gobierno, arte, entretenimiento, publicidad, educación, capacitación y presentaciones.

En la actualidad las computadoras se han convertido en una herramienta poderosa para producir imágenes, interpretar información o mejorar la calidad de visualización de las mismas en forma rápida y económica, es por ello que nos vimos en la tarea de mejorar los sistemas de información rudimentarios con los que cuenta actualmente ODAPAS, ellos al momento de recolectar la información no la procesan ni las convierten a gráficas ni imágenes solamente hacen un estudio de ellos y dan su punto de vista, nosotros mediante este proyecto mejoramos los sistemas de dar la información aplicando a estas gráficas y animaciones que nos ayudan al mejoramiento y entendimiento de los mismos, esto lleva consigo una mejora de la información para la interpretación que con frecuencia necesita ser analizada para estudiar el comportamiento de ciertos procesos.

Con este proyecto se demuestra lo importante que es la visualización mediante gráficas tridimensionales, permitiendo utilizar las capacidades de análisis visual en el humano y así, detectar los diferentes patrones de una manera mucho mas efectiva.

LISTA DE IMÁGENES

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

GLOSARIO

Archivo. Cuerpo de información codificado por medios que requieren el uso de una maquina para su tratamiento. Los archivos de computadora incluye tanto los datos almacenados en forma legible por máquina como los programas usados para procesar esos datos.

Arista. Línea que resulta de la intersección de dos superficies, considerada por la parte exterior del ángulo que forman.

Bit. Es la unidad de información más pequeña manipulada por el ordenador que adquiere un valor lógico de 1 ó 0, y está representada físicamente por un elemento como a través de un circuito eléctrico.

Escena. Comprende el espacio en que se figura el lugar de la acción a la vista del público.

Fractal. Figura geométrica con una estructura compleja y ajustable a cualquier escala.

Hardware. Palabra inglesa usada para designar la parte tangible y material de la computadora. Lo componen la máquina, conocida como fierro, el CPU (Unidad Central de Procesamiento) y todos los periféricos. Es la máquina y todas sus piezas.

Hipermedia. Integración de gráficos, sonido y video en un sistema que permite el almacenamiento y recuperación de la información de manera relacionada, por medio de referencias cruzadas; es un formato especialmente interactivo.

Hipertexto. Método de presentación de la información que permite hacer una lectura no secuencial de la misma, emula el modo en que el cerebro humano almacena y recupera la información, por medio de asociación de ideas. La información se organiza en torno a una serie de palabras.

Icono. Representación gráfica esquemática utilizada para identificar funciones o programas.

Imagen. Reproducción de un objeto formada por la convergencia de los rayos luminosos que, procedentes de él, atraviesan una lente o aparato óptico, y que puede ser proyectada en una pantalla.

Kilobyte (Kb). Unidad de medida equivalente a 1024 bytes y que se emplea para medir la capacidad de almacenamiento o memoria de una computadora (por ejemplo 640 Kb o 640 Kilobyte).

Megabyte (Mb). Es la unidad de medida equivalente a 1024 kilobytes y que se emplea para designar la capacidad de almacenamiento de las computadoras. En la actualidad, y producto del desarrollo alcanzado, existen ya computadoras con una capacidad o memoria de almacenamiento superior o equivalente a 1024 Megabytes denominada Gigabyte o Gb y también los Tetrabyte o Tb equivalentes a 1024 Gigabytes.

Memoria. Zona de almacenamiento para los datos binarios o de programas. En una computadora, la memoria se divide en dos partes: una memoria electrónica rápida e integrada a la computadora y una memoria externa, más lenta, compuesta, por ejemplo, de unidades de disco o de cintas que utilizan medios de almacenaje magnético. La memoria interna puede ser ROM, RAM etc. y la externa como disquetes.

Optimizar. Encontrar la secuencia de instrucciones que permiten lograr el objetivo deseado, empleando un mínimo de recursos (tiempo, memoria, espacio en disco, etcétera)

Ordenador. Referente a una computadora personal (PC).

Paralela. Dicho de dos o más líneas o planos: Equidistantes entre sí y que por más que se prolonguen no pueden encontrarse.

Parámetro. Dato o factor que se toma como necesario para analizar o valorar una situación.

Perpendicular. Dos líneas que se cruzan entre si y forman un ángulo recto de 90°.

Píxel. Superficie homogénea más pequeña de las que componen una imagen, que se define por su brillo y color, es un punto del monitor.

Portal. Referente a la página principal de un servidor Web.

Programa. Es un conjunto de instrucciones dirigidas a la computadora para que esta lleve a cabo una secuencia de acciones con el objetivo de realizar una o más operaciones que permitan solucionar un problema.

Sistema Operativo. Es un programa de comunicación entre el hombre y la computadora que administra y supervisa los dispositivos de hardware de una computadora, coordina la información que ingresa y sirve para controlar los programas que contenga esta.

Software. Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora, hace referencia a todos los programas de cómputo.

Trazado de Rayos. Técnica de generación de imágenes que permite la aplicación de un modelo de iluminación global para el cálculo de la iluminación en una escena.

Vector. Método de generación de imágenes que utiliza descripciones matemáticas para determinar la posición, la longitud y la dirección de las líneas que deben dibujarse.

Visualización. Técnicas para resaltar la información mediante representaciones gráficas, imágenes y animaciones.

WWW. World Wide Web, mecanismo proveedor de información electrónica para usuarios conectados a Internet. El acceso a cada sitio Web se canaliza a través del URL o identificador único de cada página de contenidos.

BIBLIOGRAFÍA

Referencia bibliográfica

- Aitken, peter. "Aprendiendo C en 21 Días". Edit. Prentice Hall. México 1998. PP 200.
- CAMEDI, Ingeniería S.A. de C.V. "Proyecto electromecánico y estructural del cárcamo de bombeo". Edit. DGCOH. México 2000. PP. 300.
- Costas, Pablo. "Persistence of Vision Ray - Tracer (Manual de Pov-Ray)". Diciembre 1997. PP. 153.
- D. Foley, James, "Introducción a la graficación por computadora". Edit Edit. Addison –Wesly Iberomaerica. Massachusetts 1993. PP. 800.
- Domínguez Mora, Ramón. "Sistema para el control y drenaje de las avenidas en el área metropolitana del valle de México". Eidt. Instituto de Ingeniería de la UNAM. México 1998. PP. 350
- Fox, David. "Gráficos animados por computadora", Edit. Mc. Graw-Hill. México 1996. PP. 525.
- García Carrillo, Rosalía. "Técnicas de Programación". Edit. McGraw Hill. México 2001. PP. 71.
- García Narro, Luis. "Manual de construcción de tanques". Edit. DGCOH – DDF. México 2000. PP. 320.
- García de Palacios Roji, Clara. "Guía Roji". 62ava. Edición 1994. México.
- Hiriart Undanavia, Humberto. "Manual de diseño de obras civiles, estructura de tanques y depósitos". Edit. CFE – Instituto de Investigaciones eléctricas. México 1969. PP. 150
- M. Chordá, Ramón. "3D Studio". Edit. Ra-ma. España 1998.
- McGhee, Terence J. "Ingeniería Ambiental y abastecimientos de agua y alcantarillado". Edit. McGraw Hill. Sexta edición 1999. PP. 300.
- Parejas, Gonzalo y de la Cruz, Jesús. "Visión por computador". Edit. Alfaomega. México 2001. PP. 792.

- Schildt, Herbert. "Programación en lenguaje C". Edit. McGraw-Hill. México 1988. PP. 278.
- Tenenbaw, Aarón. "Estructura de Datos en C". Edit. Prentice-Hall. México 1997. PP. 660.
- Vera B. Anand, "Graphical Applications". Edit. John Wiley & Sons, Inc. Clemson University 1997. PP. 635
- Winnn Lo, Rosch. "Todo sobre multimedia". Edit Prentice Hall. México 1999 PP. 700.

Referencia electrónica

- Producción de mapas y atlas de carreteras de la República Mexicana. Dirección: http://www.guiaroji.com.mx/ Diseño Gráfico: Agustín Palacios Roji R. México 2002
- Gracia, Joaquín. Dirección: http://www.webestilo.com/guia/ Enero de 2002.
- Costas, Pablo. Dirección: http://www.arrakis.es/~pcostas/povray/ Diciembre 1997.
- Castro, Antonio. Dirección: http://www.linuxfocus.org/Castellano/March1200/ 2001.
- Productos y servicios para la industria del agua en Latinoamérica. Dirección http://aguamarket.com/Diccionario/. Santiago de chile. Agosto 2002.