CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL I.P.N.

DEPARTAMENTO DE INGENIERIA ELECTRICA
SECCION DE COMPUTACION

TECNICAS PARA EL MODELADO DE FRACTALES

Tesis que presenta el Lic. LUIS ROBERTO GUTIERREZ ROMERO para obtener el grado de Maestro en Ciencias dentro de la especialidad de INGENIERIA ELECTRICA.

Este trabajo tuvó como director de tesis a :

DR. SERGIO VICTOR CHAPA VERGARA
COORDINADOR DE LA SECCION DE COMPUTACION

MEXICO, D. F.

MARZO 1993

INTRODUCCION:

Uno de los grandes problemas que ha tenido el hombre, con respecto a la graficación por computadora ha sido la dificultad que se tiene para poder dibujar objetos naturales tales como: árboles, ríos, nubes, montañas, plantas, etc. Este problema ha sido atacado con muy diversos métodos, pero ninguno de ellos ha tenido tanto éxito como el método que utiliza los **fractales**, los fractales han venido a revolucionar la graficación, ya que proveen un modelo matemático mas sencillo para la formación de algunas estructuras de la naturaleza

Este trabajo no pretende dar una visión formal de los fractales, ya que esta tesis basa su teoría en los trabajos realizados por matemáticos tales como: Peano, Mandelbrot, Barnsley, etc. El objetivo principal de esta tesis es desarrollar la programación de 4 técnicas fractales para posteriormente reunirlas en un mismo programa y realizar un editor de paisajes.

El presente trabajo de tesis tiene como objetivos:

1.- La creación de un editor de fractales, que permita interactuar con el usuario para la generación de fractales.
2.- La creación de un editor de paisajes, en donde es usuario podrá generar diferentes tipos de paisajes usando varias técnicas fractales y no fractales.
3.- La creación de fractales tridimensionales.

Este documento comprende 6 capítulos y un apéndice:

En el capítulo 1, se da una reseña histórica del desarrollo que han tenido los fractales, dando sus características y sus aplicaciones.

En el capítulo 2, se explica la primera técnica para la creación de fractales de esta tesis, la cual es denominada como Sistemas de Funciones Iteradas (IFS), mediante el uso de esta técnica se realizaran ejemplos de fractales, mostrando las ventajas y desventajas que tienen los IFS. En este capítulo se presenta tambien el editor de fractales cuya idea principal es tener una interfaz gráfica (Método del Collage) para el modelado de fractales del tipo IFS.

En el capítulo 3, se explica la técnica denominada Gramáticas de Lindermayer (Sistemas-0L). La importancia de los Sistemas-0L, se debe principalmente a su reconocimiento como un modelo formal para representar el desarrollo de plantas, dichos sistemas son gramáticas cuyas producciones se pueden aplicar en paralelo para producir fractales.

El capítulo 4, contiene otros dos métodos para la generación de fractales: Triangulación y Movimiento Browniano, en donde se presentan ejemplos de cada uno de estos métodos así como la base teórica en que se sustentan estos dos métodos.

En el capítulo 5, se describen los métodos anteriormente mencionados, pero ahora para generarlos en 3 dimensiones. (IFS, Gramáticas de Lindermayer y Movimiento Browniano)

En este capítulo 6, se dá una explicación detallada de cada una de las opciones del editor de paisajes así como la descripción de cada objeto fractal que utiliza el editor de paisajes. En este editor de paisajes por medio de fractales, se conjunto las cuatro técnicas utilizadas en los capítulos anteriores.

A través de la tesis se muestran los diversos resultados que se obtienen al aplicar las cuatro diferentes técnicas para la generación de fractales, en donde podemos apreciar las ventajas y desventajas de cada una de ellas. Debido a esto se decidió tomar las bondades de cada técnica y conjuntarlas en un solo sistema, lo cual arrojo como resultado editor de paisajes.

El apéndice de esta tesis se compone de dos partes, la primera parte corresponde al editor de fractales en donde se da un ejemplo para la creación del triángulo de Sierpinski; la segunda parte corresponde al editor de paisajes en donde se desarrolla un ejemplo de un paisaje.

En resumen, la contribución de este trabajo de tesis fue el de realizar la programación de las cuatro técnicas fractales, que fueron escritas en Lenguaje 'C' bajo el ambiente Turbo C 2.0, así como el editor de fractales tambien escrito en 'C' y el editor de paisajes. Otra contribución importante de este trabajo es que la generación de los fractales no es de una manera rígida, ya que se puede ir variando los diferentes parámetros de entrada para poder apreciar diferentes variantes de un mismo fractal. Y por último, otra contribución importante de este trabajo, es que todos los sistemas realizados en esta tesis estan orientados para ambientes PC's (desde una PC XT 286 con monitor VGA y RAM 640kg), cosa que en la mayoria de los sistemas que generan fractales requieren de sofisticado equipo de computo tales como Estaciones de Trabajo (SUN, DEC 200/25, etc.).

DEDICATORIA:

Quiero dedicar este pequeño espacio, pero muy significativo a mis seres más queridos, a mi papá el Sr. Roberto Gutiérrez Trujeque, a mi mamá Agustina Elena Romero de Gutiérrez, a mis hermanas Adriana, Flor María y a mi novia Ma. Luisa, por todos los sacrificios y esfuerzos que realizarón con todo corazón para que pudiera finalizar mis estudios de especialización. Y decirles que sin ellos no me hubiera sido posible concluir una de mis metas mas anheladas.

Tambien quiero expresar mi mas sincera gratitud a mi director de Tesis Dr. Sergio Victor Vergara Chapa, así como a mis sinodales Dr. Jaime Rangel Mondragón y el Dr. Benito Morales Luna por sus espléndidas aportaciones y consejos para la realización de este trabajo.

INDICE

1.- FRACTALES: ANTECEDENTES Y NATURALEZA

2.- SISTEMAS DE FUNCIONES ITERADAS (IFS)

3.- GRAMATICAS DE LINDERMAYER (GL)

4.- TRIANGULACION Y MOVIMIENTO BROWNIANO

5.- FRACTALES TRIDIMENSIONALES

6.- EDITOR DE PAISAJES

CAPITULO 1

FRACTALES:ANTECEDENTES Y NATURALEZA

1.1. ANTECEDENTES

A fines del siglo pasado y principios del presente (1875 - 1925) hubo un gran auge en la matemática, viejos problemas se ensayaron con métodos nuevos surgiendo de forma natural conceptos nuevos. Algunos eminentes matemáticos como :

George Cantor	[1845 - 1918]
Henri Poincare	[1854 - 1912]
Guiseppe Peano	[1858 - 1932]
David Hilbert	[1863 - 1943]
Helge von Koch	[1870 - 1924]
Waclaw Sierpinski	[1882 - 1969]
Pierre Fatou	[1878 - 1929]
Gaston Julia	[1893 - 1978]
Norbert Wiener	[1894 - 1964]

disintieron acerca de los conceptos instaurados en su disciplina que se mantenian desde la época de Euclides y meditaban profundamente acerca de problemas filosóficos de la matemática tratando de esclarecer el concepto del continum matemático o del sistema numérico real.

Las grandes realizaciones de la geometría no Euclidiana y el concepto del continum de la aritmética que subyace a la fundamentación del análisis, fueron los antecedentes y la determinación misma. A continuación explicaremos brevemente cuales fueron las aportaciones de cada uno de estos matemáticos:

CANTOR

Desde tiempos inmemoriables una de las ideas que más ha inquietado al hombre es la del infinito, aclarar su naturaleza se consideraba como una necesidad para la dignificación del mismo intelecto humano. Dentro de una esfera de especialización e interés científico, los matemáticos estuvieron tratando con este tema con la finalidad de fundamentar las matemáticas y desenredar algunos problemas de la aritmética.

En ocasión de la celebración que se tuvo en Munster en honor de Karl Weierstrass, David Hilbert preparó el tema denominado "Sobre el infinito"[8]. Expuso que hasta ese momento se tenían disputas acerca de la fundamentación del análisis, que tiempo atrás Weierstrass había presentado y se debía escencialmente a que el significado del infinito no había sido aclarado; en ese momento señalo que Cantor había llevado a cabo uno de los más profundos discernimientos acerca de la naturaleza del infinito con su teoría de Conjuntos.

En 1885 George Cantor [11] formuló la teoría de diferentes clases de infinitos conocida con el nombre de Teoría de Conjuntos, aunque atractiva y vigorosa, significaba un reto fuerte

para la intuición. A no mucho tiempo de haber salido la teoría se presentaron algunas paradojas; la situación se tornaba difícil, apenas parecia que los matemáticos se recobraban de un conjunto de paradojas - las relacionadas con la teoría de los límites en el cálculo-. Uno de los conjuntos más singulares que en ese momento fue creado es el Conjunto de Cantor del Tercio Medio (fig 1.1.), el cual ha sido uno de los ejemplos más ilustrativos de lo que es un FRACTAL, la construcción de este conjunto es relativamente simple.

Se empieza con todos los números reales que están en el intervalo [0,1], se extrae el intervalo (1/3,2/3) lo cual constituye 2/3 del intervalo original, dándonos 2 intervalos cerrados [0,1/3] y [2/3,1]. Continuando con este proceso, en cada etapa se extraerá la parte central de cualquier intervalo, este proceso puede ser aplicado una infinidad de veces (fig 1.1) E_n; los puntos que permanecen son los que forman el conjunto el Cantor, con la propiedad de que su número de elementos es infinito y la compactación de todos estos tiene longitud cero.

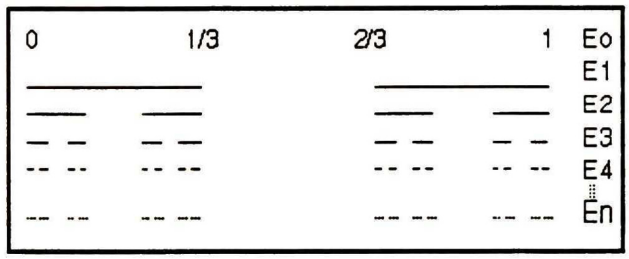


Fig 1.1 Conjunto de Cantor

Posteriormente J.M.R. Dedekind uno de los contemporáneos de Cantor dijo que "Un Sistema se dice que es infinito cuando este es similar a una menor parte del mismo (Similaridad)."

POINCARE

Henri Poincare prominente matemático francés estudió los aspectos cualitativos de la mecánica clásica en términos de la estabilidad, propiedades ergódicas y la recurrencia de órbitas; tópicos que hoy en día constituyen la teoría de la medida, topología y dinámica simbólica. El comportamiento regular en el tiempo de los fenómenos planteados con las ecuaciones determinísticas que se derivan de la segunda ley de Newton; se observó que pierden regularidad para tener un comportamiento caótico, Poincaré reconociendo el fenómeno, lo atribuyó a una sensibilidad a las condiciones iniciales y que ocurre cuando las ecuaciones que gobiernan son no lineales.

Puede suceder que para pequeñas diferencias en las condiciones iniciales se produzcan cambios grandes en el fenómeno final, un error pequeño en los antecedentes produce errores enormes en los posteriores. De esta forma la predicción de un sistema de tal naturaleza se vuelve imposible y se comporta como un fenómeno fortuito, dando como consecuencia un sistema dinámico caótico. En general, las estructuras geométricas generadas por mapas caóticos o sistemas diferenciables dinámicos son extremadamente complejas y son ejemplos claros de conjuntos fractales. Como en el ejemplo del atractor caótico del péndulo cuyo espacio fase en tres dimensiones consiste de un número infinito de capas infinitamente delgadas.

PEANO, HILBERT Y von KOCH

Una de las grandes aventuras de la matemática en este siglo fue su incursión en la geometría Fractal, algunos matemáticos iniciaron trabajos que implicaban disentir de algunos conceptos clásicos que se mantenían desde la época de Euclides. Uno de los primeros fue Moritz Pash(1843-1930) quien quizó evitar depender de las suposiciones basadas en evidencias visuales de la geometría Euclidiana para reducirla a un problema de sintaxis lógico. Giuseppe Peano fue un poco mas lejos, usando el trabajo de Pash, con una notación de lógica simbólica que el mismo inventó, obtuvo la versión de geometría de Peano completamente abstracta y que fue un cálculo de relaciones entre variables.

En 1890 Giuseppe Peano mostró cómo las matemáticas podían contradecir al sentido común, construyendo las curvas de llenado continuo (fig. 1.2).

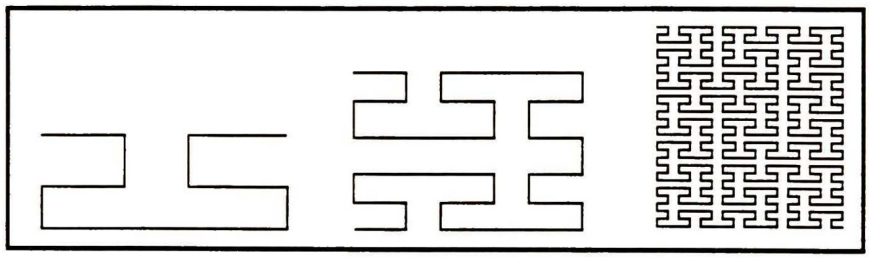
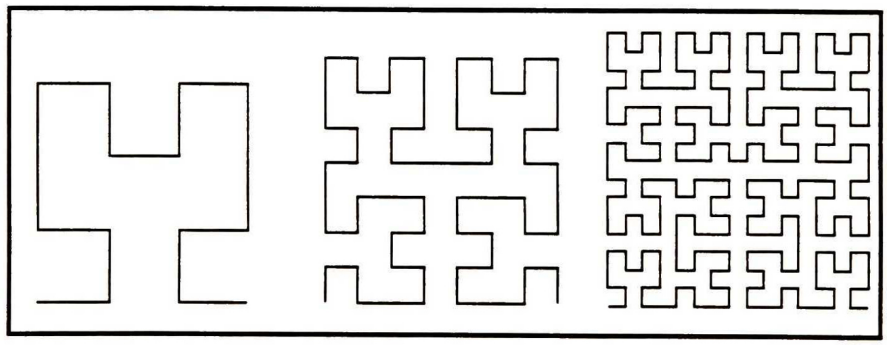


Fig 1.2 Las tres primeras iteraciones de la curva de Peano

Posteriormente David Hilbert, en 1891 publico un artículo [7], en donde propone una construcción diferente usando una base 2 en la división de los intervalos en lugar de la base 3. Esta curva es construida a partir de un cuadrado en donde cada lado del cuadrado será reemplazado por una copia del generador (fig 1.3 a), así en la primera etapa 4 subcopias serán generadas (fig 1.3 b), en la segunda etapa 16 subcopias serán generadas (fig 1.3 c) y así sucesivamente.



a b c

Fig 1.3 Las tres primeras iteraciones de la curva de Hilbert

Mas adelante Helge von Koch en 1904, generó una curva (fig. 1.4), que es considerada como la curva fractal clásica. Esta curva es construida a partir de una línea recta de longitud unitaria, la cual es dividida en tres partes iguales, después la parte central de esta línea es extraída y reemplazada por dos líneas de longitud 1/3 (fig. 1.4 E_1). Este proceso puede ser repetido una infinidad de veces y la parte central de cualquier segmento será reemplazado por dos líneas de longitud igual a un tercio de este segmento.

Nótese que los puntos (fig 1.4 E_2) de la curva que toca a la línea original E_0 son puntos del conjunto de Cantor, en cada etapa la longitud de la curva es multiplicada por 4/3 y en consecuencia la longitud de la curva de Koch es infinita. Ahora considérese el área entre la curva de Koch y la línea original E_0. En la primera etapa, un triángulo es sumado, entonces el área A es igual a: 3/36, en la siguiente etapa cuatro nuevos triángulos son sumados, sus tamaños son escalados por un factor de 1/3, así sus áreas son $(1/3)^2$ A = (1/9)A, en cada etapa, cuatro nuevos subtriángulos son sumados al área total, así el incremento en el área en cualquier etapa es 4/9 del área sumada en la etapa previa; esto nos lleva a una progresión geométrica para el área total igual a:

$$A(1 + 4/9 + (4/9)_2 + (4/9)_3 + ...) = \frac{9A}{5} = \frac{\sqrt{3}}{20}$$

de esta manera tenemos una curva de longitud infinita que encierra una área finita.

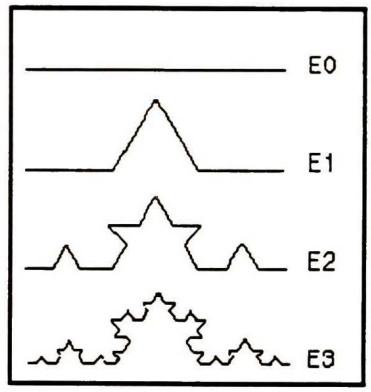


Fig 1.4 Curva de Helge von Koch

SIERPINSKI

Waclaw Sierpinski construyó varias curvas, que dieron nombre a varios objetos fractales [11], tales como la "cabeza de flecha de Sierpinski", "triángulo de Sierpinski" o la "carpeta de Sierpinski", por solo mencionar algunos.

Para construir el triángulo de Sierpinski (fig 1.5b), extraiga del triángulo original una copia invertida del triángulo, el cual estará formado por la unión de los medios puntos de los 3 lados, así 3 triángulos permaneceran y uno será removido (fig 1.5a), en la segunda etapa un cuarto del área de los tres triángulos es removido, en donde cada uno es una cuarta parte del área original, en la tercera etapa 9 triángulos son removidos de área $(1/4)^3$ del área del triángulo original. Si el área original del conjunto es A, el área removida por este proceso se obtiene de la siguiente progresión geométrica

$$A[1/4 + 3(1/4)^2 + 3^2(1/4)^3 + ..]=A [1 + 3/4 + (3/4)^2 + (3/4)^3+...]/4.$$

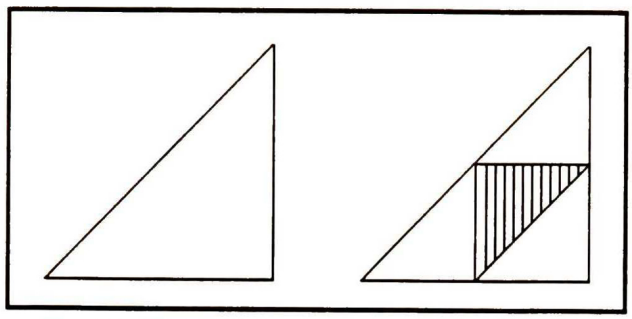


Fig. 1.5a Triángulo Original y Primera Etapa

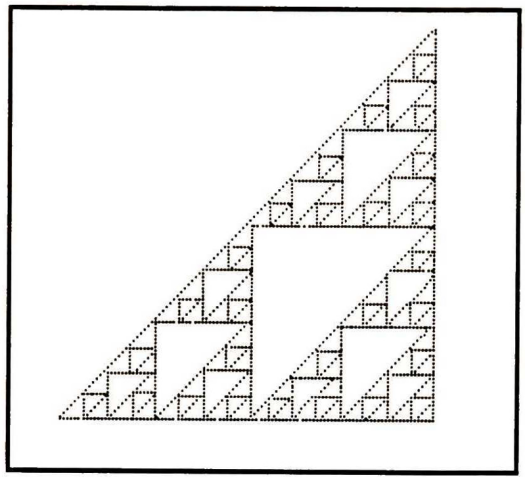


Fig 1.5b Triángulo de Sierpinski

JULIA Y FATOU

Los fractales generados por las teorías de Gaston Julia y Pierre Fatou, datan del año de 1918 y están basados en el plano complejo. Los trabajos de Julia y Fatou [11], básicamente consistieron en obtener una sucesión de puntos Z_k en el plano de los números complejos generados por la transformación $g(z) = z^2 + c$; cuando un punto inicial z_o es colocado en la transformación, el resultado de la sucesión de puntos puede comportarse de dos maneras, una que los puntos diverjan (conjuntos de escape) o que los puntos converjan a un punto fijo (conjuntos de atractores).

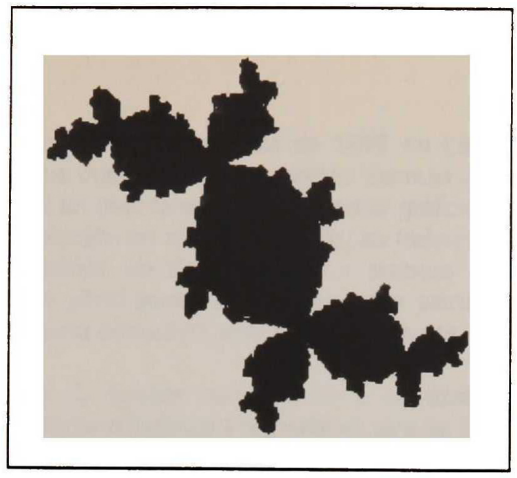


Fig. 1.6 Conjunto de Julia

NORBERT WIENER

En 1827 el botánico R. Brown [28], notó que las diminutas partículas en un líquido seguían un movimiento con trayectorias irregulares, también observó un fenómeno similar en las partículas de humo en el aire, que posteriormente fueron explicados apoyándose en el mecanismo del bombardeo molecular de las partículas. Más tarde Albert Einstein, publicó un estudio matemático de este movimiento, el cual posteriormente llevó a ganar el premio Nobel a Perrin (cálculo de los números de Avogadro).

En 1923 Norbert Wiener propuso un riguroso modelo matemático, que tenía un comportamiento aleatorio, similar al movimiento observado por R. Brown. Dicho movimiento puede ser representado por trayectorias que se desarrollan en el tiempo bajo el comportamiento observado por R. Brown; en un espacio del tiempo contra desplazamiento. Norbert Wiener establece que la partícula se puede desplazar en una unidad de tiempo una unidad de distancia con una probabilidad de un medio. Esto se hace cada vez a un tiempo nuevo tomando el nuevo punto de desplazamiento independiente del anterior (fig 1.7), dicho proceso aleatorio se denomina Proceso de Wiener y el conjunto de dichas trayectorias determinan la medida de Wiener en un espacio de funciones continuas [9].

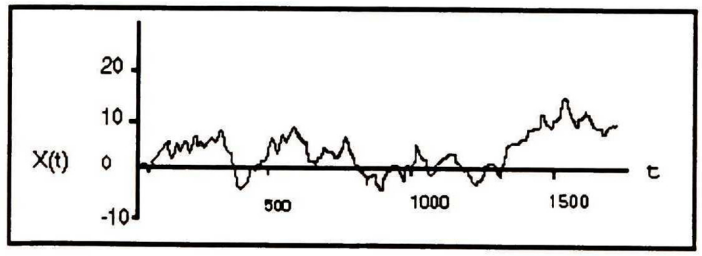


Fig 1.7 Movimiento Browniano.

Por último mencionaremos a alguien que aunque no fue matemático de profesión, jugó un papel muy importante en la revolución matemática y que dentro de esta revolución de la matemática su contribución esta mas relacionada con el arte:

ESCHER

Maurits C. Escher nació el 17 de Junio de 1898 en Leewarden, Holanda; siendo un estudiante de arquitectura, fue cuando el artista alemán Samuel J. de Mezquita lo toma bajo su tutela, ya que descubre en el un gran talento para las artes gráficas, durante el período de 1919 a 1922, Escher inicia su educación en el medio gráfico, su trabajo influenciado directamente por su forma de vida se desarrolló en dos etapas. La primera antes de 1937 en donde el reconocimiento del mundo le sirvió para dar a su obra una estructura; después de 1937 tomo como punto de partida su misma estructura para desarrollarla con motivos visibles reconocibles.

La obra de Maurits C. Escher corresponde a realidades visuales expresadas con estructuras construídas mediante contornos y superficies que se ligan (uno a uno) o (una a una) respectivamente. Los contornos se entrelazan con las superficies de acuerdo a una visión plural, vista ordenadamente y desafiando encontrar relaciones lógicas nuevas, los motivos visuales reconocibles son los que definen un esquema de inseparabilidad tanto para la estructura como para la expresión de ciertos principios matemáticos de simetría, similitudes y transformaciones que producen efectos de paradoja, ilusión o doble sentido que es una de las principales características de la obra de Escher. Por ejemplo, el cuadro Jinetes (fig 1.8) que consiste de divisiones regulares del plano con motivos de jinetes en obscuro y blanco que corren en series alternantes a la izquierda y a la derecha según la figura.

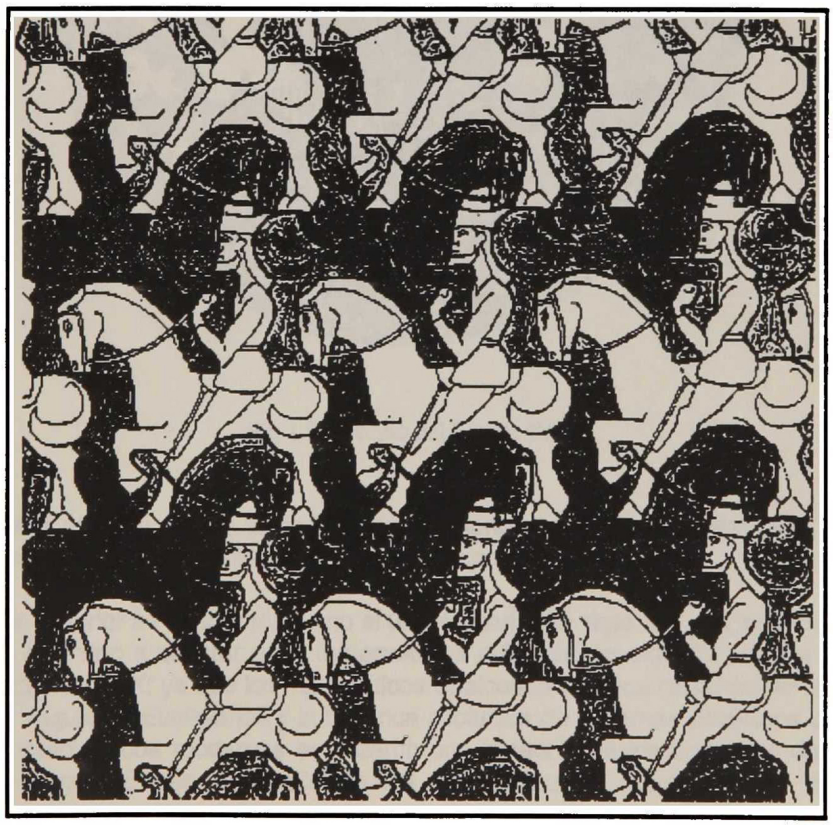


Fig 1.8 Jinetes

Quizá sea o no una casualidad pero Escher posiblemente estuvo influenciado por Henri Poincare, ya que estos dos personajes mantenían una mutua correspondencia; posteriormente Douglas Hofstadter en su libro Godel, Escher y Bach [11], expresa que Escher adopta la idea de hacer, de partes de un objeto réplicas del objeto mismo concretandose en su xilografia Copias e Igualdades: "Escher toma la idea de que parte de un objeto se empieza a copiar en el objeto mismo para irse formando a si mismo" tal como su pintura de Círculo de Límites IV (fig 1.9). Quizá Escher tuvo esta idea o no, eso nunca lo llegaremos a saber, pero una cosa si es segura, Escher fue uno de los primeros que realizó dibujos con características fractales.

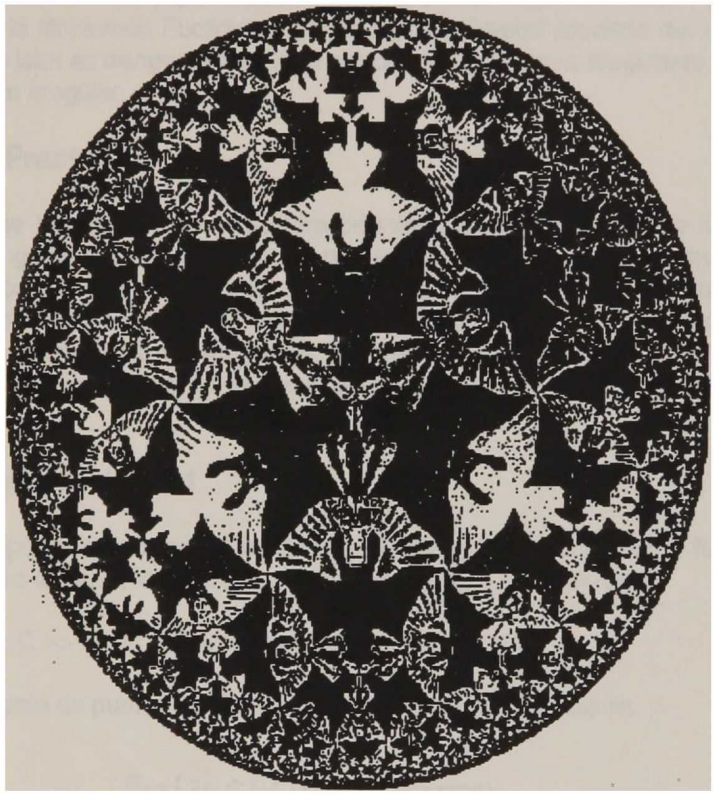


Fig 1.9 Círculo del Límite IV

Escher alcanzó en los años 50's concepciones complejas de ejemplos de geometría Fractal en ellas da implicitamente el concepto de grupo de similitudes

De lo anterior se desprende como el último cuarto del siglo XIX y el primero del XX fué el parte aguas que vino a separar a los **matemáticos clásicos** del siglo XIX de los **matemáticos modernos** del siglo XX; ya que los matemáticos clásicos tenían sus cimientos en las estructuras geométricas regulares Euclidianas y la continua evolución de sistemas dinámicos de Newton; en cambio los matemáticos modernos empezaron con la teoría de los conjuntos de Cantor y las curvas de Peano. En fin, la revolución fue forzada por el descubrimiento de estructuras matemáticas que no concordaban con los patrones Euclidianos y de Newton; estas nuevas estructuras fueron bautizadas por los matemáticos tradicionalistas con el nombre de patológicas

o como la galería de los monstruos y que actualmente se les denominó por Mandelbrot como Fractales

1.2 BENOIT B. MANDELBROT

Todos los descubrimientos antes mencionados hubieran quedado en el abandono de no ser por Benoit B. Mandelbrot del Centro de Investigaciones Thomas J. Watson en Yorktown Heights N.Y., quien interesado en las curvas de Peano y von Koch y apoyandose en la tenologia de la computación gráfica nuevos objetos que les denominó Fractales. En 1975 Mandelbrot inventa el término **Fractal**, para describir todas las curvas cuya dimensión Hausdorff-Besicovitch es mayor que la dimensión Euclidiana [28], el término **Fractal** proviene del adjetivo latín fractus, cuyo verbo en latín es frangere que indica romper en fragmentos irregulares, por lo consiguiente fractus significa irregular.

Geometría Fractal

Una de las aportaciones mas importantes de Mandelbrot fue la de crear una nueva geometría, "**la geometría fractal**", la cual puede describir figuras cuyos patrones son irregulares y fragmentados tales como nubes, montañas, árboles, etc., que no pueden ser descritas mediante la geometría Euclidiana, debido a que sus patrones no son esferas, líneas, conos, etc. De esta forma Mandelbrot planteó que la geometría Fractal describia a la naturaleza, mientras que la geometría Euclidiana es la manera natural de representar las figuras hechas por el hombre.

Conjunto de Mandelbrot

Otra aportación no menos importante que proporcionó Mandelbrot fue la caracterización de los conjuntos de Julia, los cuales se definen de la siguiente manera:

Sea f : C --> C un polinomio de grado mayor que uno.

Sea F_f el conjunto de puntos en C cuyas sucesiones divergen; esto es

$$F_f = \{ z \in C : (| f^n(z))_{n=0}^{\infty} |) \quad \text{converge} \}$$

en donde F_f se define como un conjunto de Julia completo (filled Julia set) asociado con un polinomio f; en la frontera de F_f se encuentran los llamados conjuntos de Julia. Estos conjuntos tienen una pequeña región que divide a los puntos que se van hacia el infinito y los puntos que convergen al origen, por lo tanto los conjuntos de Julia son curvas basadas en el mapeo de la función [4]:

$$z_n = z_{n-1}^2 + c$$

donde z_i y c son números complejos.

Mandelbrot desarrolló un nuevo camino para mapear estas ecuaciones: **El conjunto de Mandelbrot** (fig 1.10),

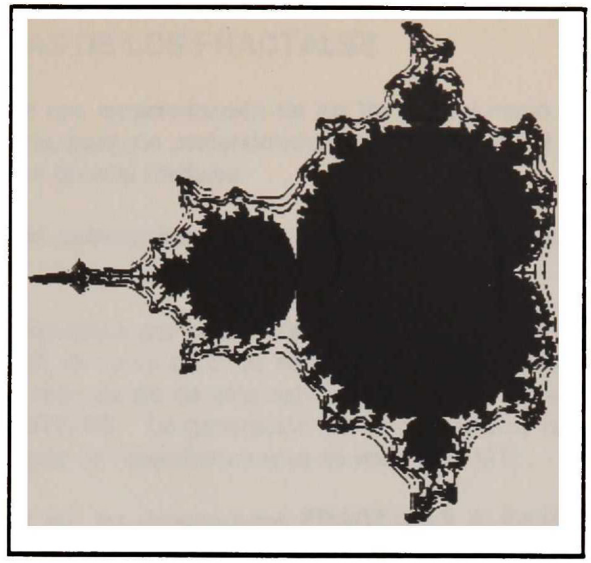


Fig 1.10 El conjunto de Mandelbrot

que son considerados como los fractales más famosos, estos conjuntos tienen la característica de tener todos los posibles conjuntos de Julia, lo cual nos da la facilidad de proporcionar los parámetros adecuados para crear un conjunto de Julia en especial (fig 1.11)

Fig 1.11 Conjunto de Julia

1.3. CARACTERISTICAS DE LOS FRACTALES

Enseguida daremos una caracterización de los fractales aunado con una definición de fractal en base a la geometría, pasando posteriormente a los capítulos 2,3 y 4 en donde veremos algunas técnicas que pueden generar fractales.

Antes de describir las características de los fractales definiremos los dos grandes grupos en los que se dividen:

1.- Aquellos que están compuestos por varias copias escaladas y rotadas de sí mismo tales como: la curva de von Koch, la curva copo de nieve o la carpeta de Sierpinski e incluso los conjuntos de Julia también caen dentro de esta categoría, este tipo fractales son denominados **FRACTALES DETERMINISTICOS.** La generación de estos fractales requiere de iteraciones (los conjuntos de Julia) o reglas de reescritura (curva de von Koch) [21].

2.- El otro tipo de fractales son los denominados **FRACTALES ALEATORIOS,** en el cual se puede encontrar como una de sus características principales la aleatoriedad, un ejemplo clásico de este tipo de fractales es el movimiento Browniano [21].

A continuación daremos algunos puntos que nos ayudarán a reconocer lo que es un **fractal**:

1.-Las curvas fractales tienen una dimensión diferente que la dimensión geométrica Euclidiana.
2.-Los fractales tienen un perímetro infinito conteniendo una área finita.
3.-Los fractales pueden ser definidos por ecuaciones matemáticas simples, como por ejemplo la ecuación de los conjuntos de Julia ($Z_n = Z_{n-1}^2 + C$)
4.-Los fractales tienen la característica de ser autosimilares.

1.4. APLICACIONES DE LOS FRACTALES

Los fractales han tenido una gran apogeo en los últimos años, ya que se han encontrando un amplio campo de aplicaciones dentro de las siguientes áreas: El Arte [35], La Medicina [20], La Biología [11], La Música [36], La Cinematografía [35], etc.

CINEMATOGRAFIA

En 1972, Benoit B. Mandelbrot con Hirsh Lewitan hicieron una escena para la creación de un conjunto de galaxias fractales usando el método llamado imagen del Universo [35]; en 1975 con Sig Handelman hicieron un paisaje el cual emerge lentamente de las profundidades rotando majestuosamente y regresando lentamente al agua. La potencialidad de los fractales de Voss en la creación de los paisajes fue muy grande, así que los fractales fueron introducidos con cierta aceptación a las películas.

Los pilares de la extensión de los fractales hacia la cinematografía fueron Alan Fournier, Don Fusell y Loren Carpenter [17], ya que hicieron uno de los más grandes trabajos de los fractales en la cinematografía, este trabajo fue hecho en la película "Viaje a las Estrellas II (La furia de Khan)", en esta película se presentan unas secuencias generadas por computadora, en donde los paisajes fractales se han hecho clásicos en la comunidad de graficación por computadora. La escena fractal más conocida que se ha usado en las películas corresponde a

Viaje a las Estrellas II, en donde se tiene la secuencia de transformación del planeta Génesis; posteriormente Digital Productions incluyó masivamente paisajes de fractales en la película el Ultimo Guerrero (The Last Starfighter).

BIOLOGIA

Muchos métodos han sido descritos en graficación por computadora para la generación de objetos que reensamblan a objetos biológicos, un ejemplo de este es el que realizó Kawaguchi [24], el cual creo un método para la creación de imágenes de corales y conchas. Otro método que fue desarrollado para describir objetos biológicos es el de Gramáticas de Lindermayer [36] y por último otro ejemplo en donde se aplicaron los fractales a la biología fue en los modelos de crecimiento de objetos biológicos [30].

RECONOCIMIENTO DE IMAGENES

En el área de procesamiento de imágenes los fractales ha sido aplicados para describir la figura de objetos naturales complejos e imágenes de superficies tales como: fotografías áreas, el despliegue de imágenes por satélite [32] y en la compactación de imágenes por satélite [11].

GEOLOGIA

En la geología, el uso de los fractales no ha sido la excepción, ya que en Denver (E.U.), el geólogo Christopher C. Barton, esta usando a la geometría fractal para tratar de determinar, cuanto petroleo y gas natural permanece en la tierra [23].

MUSICA

La música Fractal es todavía una rama no muy conocida, sin embargo existen algunos autores [26], que están empezando a experimentar con la aplicación de los fractales a la música, el resultado que se ha obtenido hasta ahora no ha tenido gran relevancia, pero quizá en unos cuantos años, estaremos escuchando alguna música fractal.

CAPITULO 2

SISTEMAS DE FUNCIONES ITERADAS (IFS)

La factibilidad de usar Sistemas de Funciones Iteradas (IFS) en computadora, data del año de 1985 [13]; cuando Demko, Naylor y Hodges, orientan su trabajo al desarrollo de los IFS para la producción de imágenes tales como: nubes, horizontes y paisajes entre otros.

El uso de la geometría fractal, (determinística o aleatoria) hacia la modelación de escenas naturales y objetos, ha sido investigado por un gran número de autores: Mandelbrot [28], Kawaguchi [24], Oppenheimer [34], Fournier et al[17], Smith [38], Miller [31] y Amburn et al[1], cuyas investigaciones han sido consultadas y utilizadas en este trabajo como la metodología fundamental en el desarrollo algorítmico del editor de fractales. No obstante, este se caracterizó por un simple esqueleto determinístico, el cual aparentemente puede alcanzar un número ilimitado de objetos.

La dirección propuesta para modelar y ejecutar los problemas se hará por medio de dos nuevos métodos de IFS sugeridos por M. Barnsley [4,6,35]:

1.- El Método del Collage, que sugiere un método interactivamente geométrico para encontrar los códigos de los Sistemas de Funciones Iteradas (IFS).

2.- Un algoritmo de iteración aleatoria, para desplegar la geometría de las imágenes, teniendo previamente sus códigos.

El Método del Collage provee un medio para modelar objetos bidimensionales usando IFS, el cual tiene como entrada una aproximación poligonizada y como salida el código de las transformaciones afines.

Esta innovación contrasta con el uso de algoritmos aleatorios para producir terrenos [31], procedimientos estocásticos para producir nubes y texturas [28], y modelos de crecimiento aleatorio para producir plantas ([2, 24, 34]). En todos estos casos, el producto final depende de la secuencia de números aleatorios usados durante el cálculo en el algoritmo; lo cual contrasta con el método de IFS ya que pequeños cambios en los parámetros de entrada producen pequeños cambios en la imagen de salida.

2.1 EL JUEGO DEL CAOS

Supongamos que tenemos tres puntos etiquetados como: águila, sol y cara los cuales están marcados en una hoja; asumamos también que tenemos una moneda especial la cual tendrá tres posibilidades de caer (águila, sol o cara). Para poder iniciar el juego marquemos un punto en el papel (dentro del triángulo) y etiquetemoslo como z_1. Si lanzamos la moneda y esta cae águila entonces marquemos un nuevo punto z_2 entre la mitad de z_1 y el punto etiquetado como águila, lanzando de nuevo la moneda y esta cae sol, marquemos un nuevo punto z_3 entre

z_2 y el punto etiquetado como sol, siguiendo de esta manera hasta el n-ésimo paso se tendrán marcados los puntos en el papel $z_1, z_2, \ldots z_n$; con esto se tendrá una sucesión de puntos (fig. 2.1).

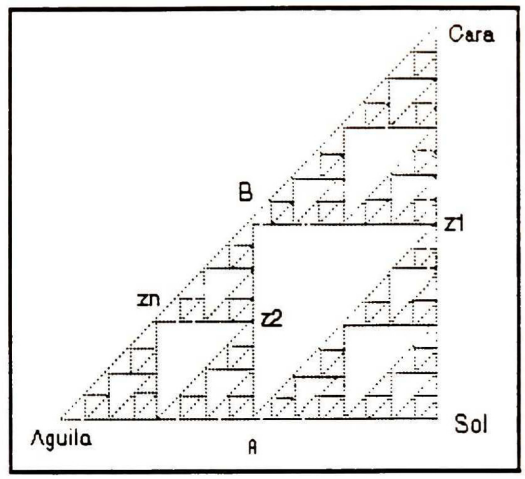


Fig. 2.1 El Juego del Caos.

Si a esta imagen la llamamos P, nótese que P es la unión de tres copias de sí mismo; de hecho

$$P = w_1(P) \cup w_2(P) \cup w_3(P) \qquad (2.1)$$

donde por ejemplo, w_1 es una transformación afín que toma el triángulo cuyos vértices son águila, sol y cara y los convierte en el triángulo con vértices águila, A y B respectivamente.

Generalizando el juego del caos, se tiene que $w_1, w_2, \ldots w_n$ son transformaciones las cuales toman un punto z en el plano $z \in \mathbf{R}^2$ y lo mueven a un punto $w_i(z) \in \mathbf{R}^2$, si cada transformación cumple con la siguiente condición:

$$d(w_j(z_1), w_j(z_2)) < k * d(z_1, z_2)$$

para algún $0 \leq k < 1$, entonces se dice que tal transformación es un mapeo contractivo.

Un IFS en un espacio bidimensional tiene dos distintas componentes, la primera consiste de un conjunto de N transformaciones donde N es un número entero:

$$\{w_1, w_2, \ldots\ldots w_n\}$$

de $\mathbf{R}^2 \rightarrow \mathbf{R}^2$ y la segunda es un conjunto de probabilidades

$$\{p_1, p_2, \ldots\ldots p_n\}$$

donde cada $p_i > 0$ y

$$p_1 + p_2 + \ldots\ldots + p_n = 1.$$

aquí p_i puede ser considerado como pesos relativos de cada una de las transformaciones; por lo tanto un código IFS es:

$$\{ W_n, P_n : n=1,2,..,N \}$$

2.2 OBTENCION DE LOS CODIGOS IFS

Para dar la idea de la obtención de códigos IFS, empezaremos dando un ejemplo ilustrativo, el cual consiste en copiar los detalles de la hoja grande a la hoja chica (fig. 2.2).

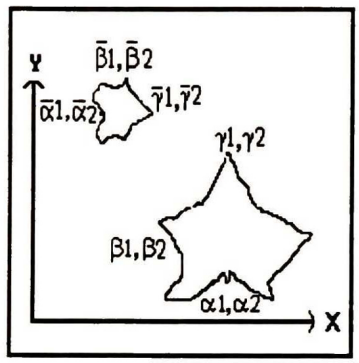


Fig 2.2 Obtención de una transformación afín.

Ahora si deseamos encontrar los números reales a, b, c, d, e y f (códigos de las transformaciones afines), tenemos que resolver la siguiente transformación

$$w\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} ax + by + e \\ cx + dy + f \end{bmatrix}$$

que tiene la propiedad w(hoja grande) ≈ w(hoja pequeña).

Empezamos introduciendo los ejes x,y (fig. 2.2.), después marcamos tres puntos diferentes en la hoja grande (los que se deseen) y determinamos sus coordenadas:

$$(\alpha 1,\alpha 2),\ (\beta 1,\beta 2) \text{ y } (\gamma 1,\gamma 2)$$

se marcan los puntos correspondientes en la hoja pequeña, (suponiendo que una oruga no se ha comido a la hoja) y determinamos sus coordenadas, digamos:

$$(\bar{\alpha}_1,\bar{\alpha}_2),\ (\bar{\beta}_1,\bar{\beta}_2) \text{ y } (\bar{\gamma}_1,\bar{\gamma}_2)$$

respectivamente.

Entonces a,b y e son obtenidas resolviendo las tres ecuaciones lineales:

$$\begin{aligned} \alpha 1\, a + \alpha 2\, b + e &= \bar{\alpha}_1 \\ \beta 1\, a + \beta 2\, b + e &= \bar{\beta}_1 \\ \gamma 1\, a + \gamma 2\, b + e &= \bar{\gamma}_1 \end{aligned} \qquad \text{Ecuación (2.2)}$$

mientras que c, d y f satisfacen

$$\begin{aligned} \alpha 1\, c + \alpha 2\, d + f &= \bar{\alpha}_2 \\ \beta 1\, c + \beta 2\, d + f &= \bar{\beta}_2 \\ \gamma 1\, c + \gamma 2\, d + f &= \bar{\gamma}_2 \end{aligned} \qquad \text{Ecuación (2.3)}$$

2.3 CARACTERISTICAS DE LOS IFS

Los códigos de los IFS son transformaciones afines w: $\mathbf{R}^2$ -> $\mathbf{R}^2$ en el espacio bidimensional $\mathbf{R}^2$ y están definidos como:

$$w\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y + b_1 \\ a_{21}x + a_{22}y + b_2 \end{bmatrix}$$

donde los a_{ij}'s y b_i's son constantes reales. Si A denota la matriz (a_{ij}), b denota el vector $(b_1,b_2)^t$, donde t es el transpuesto y $\vec{x}$ denota el vector $(x_1, x_2)^t$, entonces escribiremos:

$$w(\vec{x}) = A\vec{x} + \vec{b}$$

La transformación afín se especifica por seis números reales. Dada una transformación afín, uno puede siempre encontrar un número s no negativo tal que:

$$|| w(\vec{x}) - w(\vec{y}) || < s || \vec{x} - \vec{y} || \quad \text{Para todo } \vec{x}, \vec{y}$$

donde el número mínimo s es conocido como la constante de Lispchitz para w y tal transformación se clasifica de la siguiente manera de acuerdo al valor de S:

$$S \begin{cases} < 1 & \text{Contractiva} \\ = 1 & \text{Simétrica.} \\ > 1 & \text{Expansiva.} \end{cases}$$

Si $\{W_n,P_n: n=1,2,...,N\}$ es un código IFS, entonces por el Teorema de Barnsley y Elton ([5, 15]), hay un único objeto geométrico asociado, denotado por A(un subconjunto de $\mathbf{R}^2$) llamado el atractor del IFS y una sola medida denotada por μ; de esta forma el modelo fundamental asociado con el código IFS es una dupla (A,μ) donde A es un atractor y μ una medida.

2.4 METODO DEL COLLAGE

En un código IFS $\{w_n,p_n:n=1,2,...N\}$, los w_n's determinan la geometría del modelo principal, es decir la estructura del atractor, mientras que las p_n's son las que proporcionan con

que frecuencia se visitará tal transformación. Describimos un algoritmo de interacción geométrica bidimensional para determinar los w_n's correspondientes de cada modelo deseado; este algoritmo tiene las bases matemáticas en el Teorema del Collage [6].

Teorema del Collage:

Sea $\{w_n, p_n : n=1,2,..N\}$ un código de mapeos afines IFS contractivos, en donde $s<1$ (constante de Lipschitz), $\varepsilon>0$, T es un subconjunto cerrado dado en $\mathbf{R}^2$, y los mapeos w_n han sido escogidos tal que

$$h\,|\,T, \bigcup_{n=1}^{N} w_n\,(T)\,| < \varepsilon$$

entonces

$$h(T, A) < \varepsilon / 1\text{-}s$$

donde A denota el atractor de IFS.

El método del Collage empieza con una imagen T, la cual esta delimitada por la ventana V [0,1] x [0,1], T puede ser cualquier imagen digitalizada o una poligonización de la imagen. Una transformación afín $w_1(\vec{x})=A^{(1)}\,\vec{x} + \vec{b}^{(1)}$ se introduce con los coeficientes inicializados

$$a_{11}^{(1)} = a_{22}^{(1)} = 0.25,\ a_{12}^{(1)} = a_{21}^{(1)} = b_2^{(1)} = 0$$

ahora teniendo $w_1(T)$, el usuario ajustará interactívamente $a_{i,j}^{(1)}$, así que la imagen $w_1(T)$ es trasladada y rotada en la pantalla, la meta del usuario es transformar $w_1(T)$ hasta que esta cubra parte T.

Es importante señalar que las dimensiones de $w_1(T)$ deben ser más pequeñas que las de T, para asegurar que w_1 sea una contracción; una vez que $w_1(T)$ está posicionada se fija y una nueva subcopia de la imagen $w_2(T)$ se introduce, entonces w_2 es ajustada interactívamente hasta que $w_2(T)$ cubra un subconjunto de T que difiere de $w_1(T)$. El traslape entre $w_1(T)$ y $w_2(T)$ es permitido, pero para tener una mejor aproximación al objeto esta deberá de ser lo más pequeño posible.

De esta manera el usuario determina un conjunto de transformaciones afines contractivas $\{w_1, w_2, ..., w_N\}$ con la propiedad:

$$\bar{T} = \bigcup_{n=1}^{N} w_n\,(T)$$

con N lo más pequeña posible.

El algoritmo es ilustrado en la fig. 2.3, la cual muestra una poligonización T de una hoja, en este caso T sera cubierta por cuatro transformaciones afines.

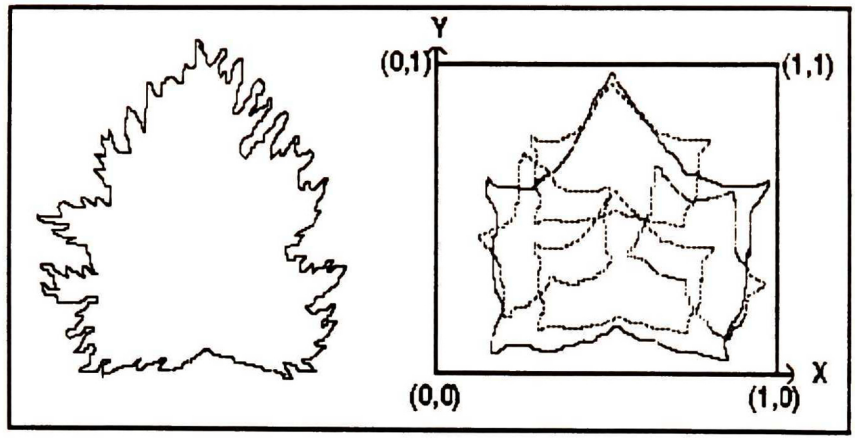


Fig 2.3 Método del Collage.

```
/* Algoritmo del Collage  */
.....
Pide_Punto(&bx,&by,2); /* Obtiene la Base de la nueva figura */
ba1.x = bx;                          /* que se va a fractalizar                    */
ba1.y = by;
Pide_Punto(&bx,&by,2);
ba2.x = bx;
ba2.y = by;
Temporal = Ancla;
A.x = Ancla -> Xr;
A.y = Ancla -> Yr;
Temporal = Temporal -> next;
B.x = Temporal -> Xr;
B.y = Temporal -> Yr;
Temporal = Ancla;
for (i = o ; i < ver;; i ++) /* Aplica la Similaridad a la Base dada*/
 {
        Ps.x = Temporal ->Xr;
        Ps.y = Temporal ->Yr;
        Similar(Ps, ba1, ba2, &ax, &ax1);
        Q[j][i].x = ax;
        Q[j][i].y = ax1;
        Temporal = Temporal -> next;
 }
....
....
```

Rec. 2.1 Algoritmo del Método del Collage

2.5 METODO DE ITERACION ALEATORIA

El método de iteración aleatoria necesita como entrada los códigos IFS (w_n,p_n:n=1,2,...,N), obtenidos al aplicar el método del collage, este método calcula las imágenes del IFS definidas previamente. Un camino aleatorio en $\mathbf{R}^2$ es generado a partir del código IFS (Recuadro 2.1).

Un punto inicial(x_0,y_0) necesita fijarse (por lo general siempre se fija x_0=0 y y_0=0), si elegimos (x_0,y_0) como un punto fijo de w_1 sabiendo a priori que (x_0,y_0) pertenece a la imagen. Esto se obtiene resolviendo la ecuación lineal

$$\begin{vmatrix} Xo \\ Yo \end{vmatrix} - A \begin{vmatrix} Xo \\ Yo \end{vmatrix} = \begin{vmatrix} b1 \\ b2 \end{vmatrix}$$

En resumen el algoritmo de **iteración aleatoria** se realiza de la siguiente manera:

1.- Se selecciona un punto inicial (0,0).
2.- Se selecciona una transformación afín w_i con una cierta probabilidad p_i
3.- Se aplica esta transformación con los valores obtenidos previamente, para obtener nuevos puntos.
4.- Se pintan los puntos obtenidos.
5.- Se aplican los pasos del 2 al 4 hasta un cierto número de iteraciones inicialmente definidos.
6.- Termina.

A continuación mostraremos en el siguiente recuadro (2.2), parte del código del método de iteración aleatoria.

```
/* Algoritmo de Iteración Aleatoria */
x= 0
y= 0;
for i = 1 to Num do  /* Numero de iteraciones  */
  r = random () % No_iter;
  xini = a(r) * x  + b(r)* y + e(r); /* Aplica la transformación */
  y = c(r) * x + d(r) * y + f(r);
  x = xini;
  setcolor(r);
  Moveg (x,y);
  Drawg (x,y);  /* Pinta el punto  encontrado */
```

Rec. 2.2 Algoritmo de Iteración Aleatoria

2.6 VENTAJAS Y DESVENTAJAS DE LOS IFS

Las **ventajas** que presenta el algoritmo de IFS son bastantes:

1.- La información para la modelación de estos objetos puede ser almacenada en una pequeña base de datos.
2.- El algoritmo de IFS puede ser implementado en sistemas paralelos.
3.- El método es relativamente fácil de entender.
4.- Se puede asignar diferentes colores a cada una de las transformaciones para dar una apariencia más real al objeto.
5.- Los objetos que se pueden modelar son prácticamente ilimitados.
6.- La modelación de los objetos es rápida.

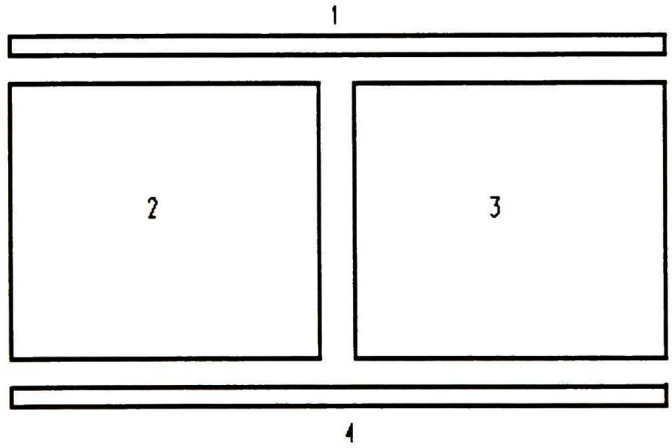
Como en todos los algoritmos hay algunos inconvenientes; el algoritmo de IFS no es la excepción ya que tiene las siguientes **desventajas**:

1.- Para poder realizar una imagen de alta calidad es necesario un número grande de iteraciones (100,000)

2.7 EDITOR DE FRACTALES

2.7.1 DESCRIPCION GENERAL

El editor de fractales se hizo con la idea de tener una interfaz gráfica para poder modelar fractales del tipo IFS; ya que este editor nos da la facilidad de poligonizar al objeto que deseamos encontrar sus códigos IFS y después ir formando al objeto que deseamos fractalizar. Una vez obtenido los códigos del objeto poligonizado el editor permite realizar la ejecución del método de iteración aleatoria para graficar al objeto que se fractalizo y ver cual es el resultado. El **editor de fractales** consta de cuatro ventanas:



1.- Ventana del Menú Principal.
2.- Ventana de la Poligonización de la Imagen.
3.- Ventana de Interacción.
4.- Ventana de Diálogo.

El editor de fractales cuenta con un manejador para el menú y para la pantalla. Una característica importante del menú es la elección de las opciones condicionadas, ya que si la opción EDIT no ha sido seleccionada previamente las demás opciones (excepto QUIT) no pueden ser elegidas; otra característica que se debe tener en cuenta es que las coordenadas del origen de la pantalla están cambiadas, ya que el origen se encuentra en la parte inferior

izquierda del monitor y no en la parte superior izquierda del monitor como regularmente se maneja.

2.7.2 MENU PRINCIPAL

El editor de los fractales cuenta con las siguientes opciones:

EDIT

La opción EDIT es la que permite editar la poligonización del fractal que se desee; esto se realiza gracias a que las aristas que se irán marcando son las que definirán al objeto en cuestión.

SIMIR

La opción SIMIR nos permite crear las copias autosimilares que conformarán la figura poligonizada que se dio, esto se hace gracias a que esta opción rota, escala y traslada el objeto poligonizado hacia donde se le indique (respecto a la base) en la pantalla. Una vez hecho este proceso, se realizarán los cálculos para determinar los códigos de las transformaciones afines de la figura que se proporcionó, para que sirvan de entrada al programa que crea las figuras fractales(EXEC). Estos códigos se obtienen resolviendo los sistemas (2.1 y 2.2), que resulten de seleccionar las aristas de la figura poligonizada.

SAVE

La opción SAVE nos guardará en un archivo las constantes de las transformaciones afines que se obtuvieron de la figura poligonizada.

EXEC

Crea la figura fractal que se poligonizó, por medio del algoritmo de iteración aleatoria, este algoritmo realiza una serie de pasos previos para generar la imagen fractal:

a) Se ejecutará el algoritmo de iteración aleatoria pero sin pintar ningún punto, ya que esta primera corrida de este algoritmo nos permitirá conocer los máximos y mínimos de esta figura fractal.

b) Se ejecuta el algoritmo de iteración aleatoria, pero pintando los puntos correspondientes de la figura, luego de escalar y definir el puerto de visión

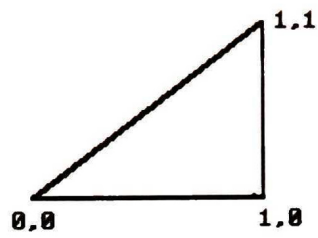
QUIT

Esta opción nos permite salir del programa.

2.8 EJEMPLOS

A continuación daremos un ejemplo de la manera en que se obtienen los códigos de las transformaciones para un fractal dado, el ejemplo es el triángulo de Sierpinski:

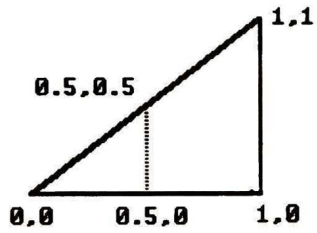
1.- Damos una imagen poligonizada del triángulo de Sierpinski



La región en donde se poligoniza la imagen, es una región cuadrada teniendo como extremos las coordenadas (0,0) y (1,1).

2.- Se crean tres copias autosimilares del triángulo de Sierpinski para llenar toda la imagen que se poligonizó.

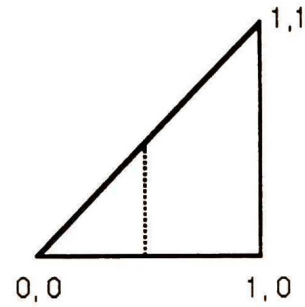
a) La primer copia autosimilar se colocará en los vértices (0,0), (0,0.5), (0.5,0.5) de la imagen poligonizada.



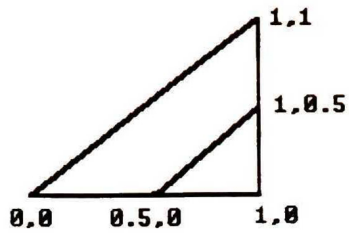
El sistema que se genera con estos vértices es:

$$0a + 0b + e = 0 \rightarrow e = 0.0$$
$$1a + 0b + e = 0.5 \rightarrow a = 0.5$$
$$1a + 1b + e = 0.5 \rightarrow b = 0.0$$

$$0c + 0d + f = 0.0 \rightarrow f = 0.0$$
$$1c + 0d + f = 0.0 \rightarrow c = 0.0$$
$$1c + 1d + f = 0.5 \rightarrow d = 0.5$$



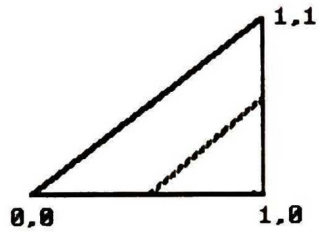
b) La segunda copia autosimilar se colocará en los vértices (0.5,0), (1,0) y (1,0.5) de la imagen poligonizada



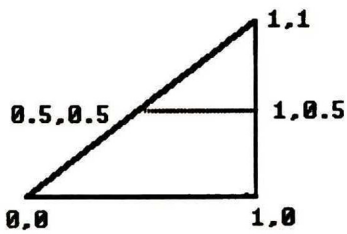
El sistema que se genera con estos vértices es:

0a + 0b + e = 0.5 -> e = 0.5
1a + 0b + e = 1.0 -> a = 0.5
1a + 1b + e = 1.0 -> b = 0.0

0c + 0d + f = 0.0 -> f = 0.0
1c + 0d + f = 0.0 -> c = 0.0
1c + 1d + f = 0.5 -> d = 0.5



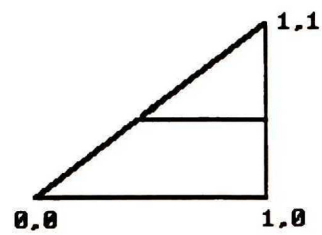
c) La última copia autosimilar se coloca en los vértices (0.5, 0.5), (1,0.5) y (1,1) en la imagen poligonizada.



El sistema que se genera con estos vértices es:

0a + 0b + e = 0.5 -> e = 0.5
1a + 0b + e = 1.0 -> a = 0.5
1a + 1b + e = 1.0 -> b = 0.0

0c + 0d + f = 0.5 -> f = 0.5
1c + 0d + f = 0.5 -> c = 0.0
1c + 1d + f = 1.0 -> d = 0.5



Por lo tanto los códigos de las transformaciones afines del triángulo de Sierpinski quedan determinados. Por último queda por especificar las probabilidades de cada una de estas transformaciones, las cuales se definen como:

$$P_i = a_i d_i - b_i c_i / (\Sigma \mid a_k d_k - b_k c_k \mid)$$

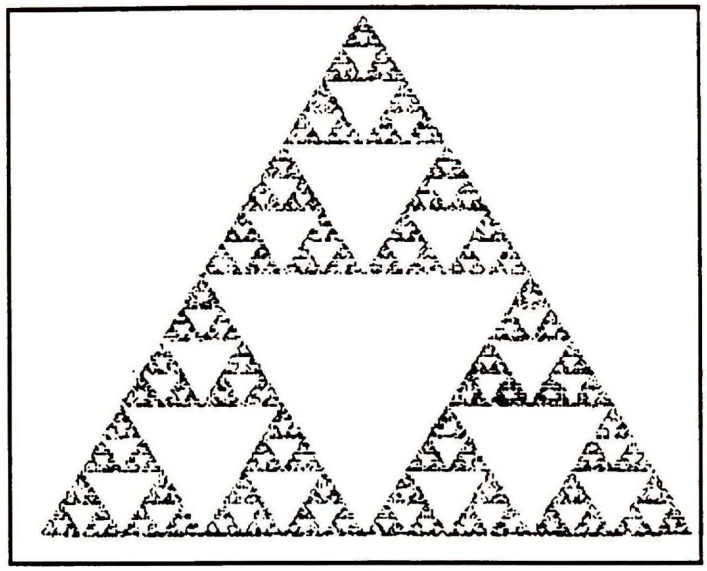
así que las probabilidades del triángulo de Sierpinski son:

$p_1 = 0.25 / 0.75 = 0.333$
$p_2 = 0.25 / 0.75 = 0.333$
$p_3 = 0.25 / 0.75 = 0.333$

A continuación se presentan los siguientes ejemplos : Triángulo de Sierpinski, Arbol, Arbol de Cantor, Helecho.

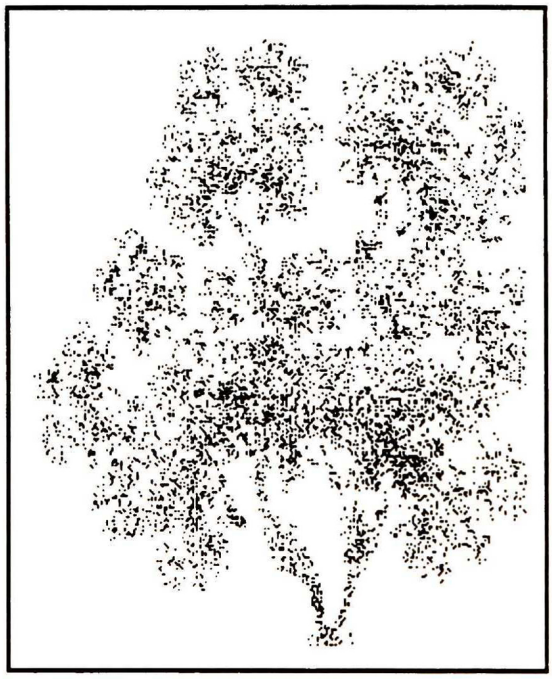
Ejemplo 1 TRIANGULO DE SIERPINSKI
Códigos

	W1	W2	W3
a	0.5	0.5	0.5
b	0.0	0.0	0.0
c	0.0	0.0	0.0
d	0.5	0.5	0.5
e	0.0	0.5	0.5
f	0.0	0.0	0.5
P	0.33	0.33	0.33



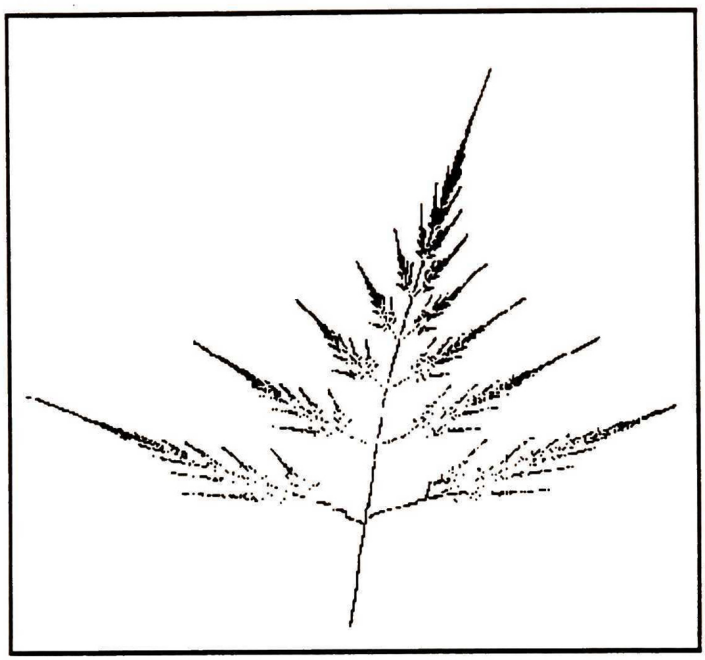
Ejemplo 2 ARBOL
Códigos

	W1	W2	W3	W4
a	0.0	0.1	0.42	-0.42
b	0.0	0.0	-0.42	0.42
c	0.0	0.0	0.42	-0.42
d	0.5	0.1	0.42	0.42
e	0.0	0.0	0.0	0.0
f	0.0	0.2	0.2	0.2
p	0.04	0.1	0.4	0.4



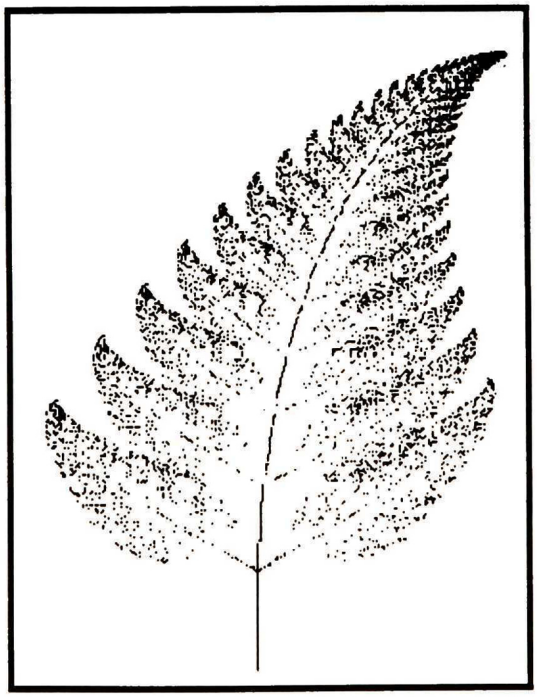
Ejemplo 3 ARBOL DE CANTOR
Códigos

	W1	W2	W3
a	0.33	0.33	0.66
b	0.0	0.0	0.0
c	0.0	0.0	0.0
d	0.33	0.33	0.66
e	0.0	1.0	0.5
f	0.0	0.0	0.5
P	0.32	0.32	0.34



Ejemplo 4 HELECHO
Códigos

	W1	W2	W3	W4
a	0.0	0.2	-0.15	0.85
b	0.0	-0.26	0.28	0.04
c	0.0	0.23	0.26	-0.04
d	0.16	0.22	0.24	0.85
e	0.0	0.0	0.0	0.0
f	0.0	0.2	0.2	0.2
P	0.01	0.06	0.07	0.851



CAPITULO 3

GRAMATICAS DE LINDERMAYER

En 1968 el biólogo Danés Aristid Lindermayer, introdujo un nuevo tipo de reglas de reescritura, posteriormente denominado Sistemas-L, con el propósito de modelar el crecimiento de los organismos vivos, en particular los patrones de las ramas en las plantas.

Las gramáticas de Lindermayer o Sistemas-L encontraron un número creciente de aplicaciones en graficación por computadora; siendo dos áreas las principales: la generación de fractales y la modelación realista de plantas [36]. Además de las dos aplicaciones anteriores, las gramáticas de Lindermayer han servido para modelar estructuras de arte Hindú así como algoritmos para componer música [26].

3.1 SISTEMAS DE REESCRITURA.

Antes de proceder con los detalles, nos centraremos en el concepto de reescritura el cual es el núcleo de los Sistemas-L. La idea básica es definir objetos complejos por medio del reemplazamiento sucesivo de partes de un objeto simple inicial usando reglas de reescritura o producciones.

El ejemplo clásico de un objeto gráfico definido en términos de las reglas de reescritura es la curva "copo de nieve" propuesto por von Koch en 1905. La construcción la plantea Mandelbrot [28] de la siguiente manera: Se empieza con dos figuras: un iniciador y un generador (fig 3.1), en cada etapa de la construcción se irá reemplazando cada línea con una copia del generador, reduciendo y desplazándose hasta llegar al punto en donde se inició

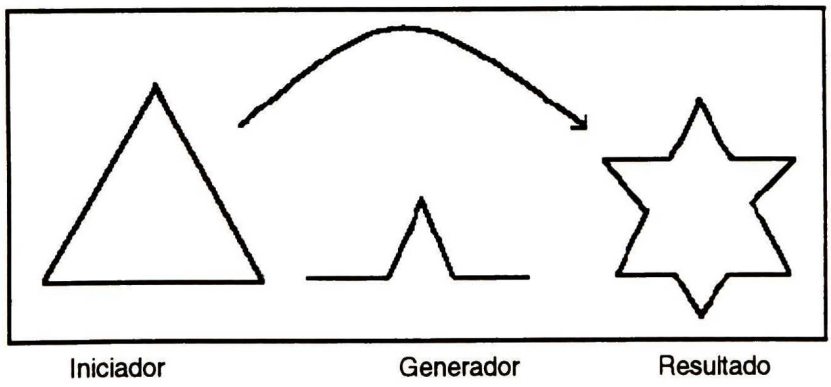


Fig. 3.1 Construcción de la curva copo de nieve.

un mecanismo similar del arreglo de reescritura es el que se presenta en el mecanismo del juego de la vida presentado por Conway [19].

La primera definición formal de tales sistemas fue dada en el inicio de este siglo por Thue, posteriormente un amplio interés fue presentado por Chomsky en las postrimerías de 1950 con sus gramáticas formales [10].

Pocos años después Backus y Naur introdujeron la notación de "reescritura-basada en el orden" para describir formalmente características sintácticas de los lenguajes de programación (ALGOL-60) [3]. La equivalencia de la forma de Backus-Naur (BNF) y las clases de gramáticas libres de contexto de Chomsky fueron pronto reconocidas. El centro de la atención fueron los conjuntos de cadenas (llamados lenguajes formales).

Las innovaciones antes mencionadas en la teoría de lenguajes formales rompió con la idea de usar cadenas de caracteres para describir figuras; la meta inicial fue el reconocimiento de objetos, tales como cartas manuscritas [33] o cromosomas [27], mediante la codificación de sus imágenes utilizando cadenas, entre la variedad considerada de representación de imágenes está el código de la cadena propuesto por Freeman [25], para describir formas geométricas precisas, lo cual propició una investigación de las relaciones entre imágenes y clases de lenguajes de una manera formal; tal estudio fue llevado a cabo por Feder [16].

La introducción de los Sistemas-L revivió el interés en la representación de imágenes usando cadenas de caracteres. Los Sistemas-L fueron reconocidos como un modelo de representación en el desarrollo de plantas, los primeros resultados fueron publicados por Frijters y Lindermayer en 1974 [18], y Hogeweg y Hosper [22]; en ambos casos los Sistemas-L fueron usados primordialmente para determinar el crecimiento topológico del modelado de plantas; el aspecto geométrico, tales como la longitud de las líneas y ángulos de crecimiento fueron sumados en fases posteriores, los resultados de Hogeweg y Hosper fueron subsecuentemente extendidos por Smith [38], quién demostró la potencialidad de los Sistemas-L para la síntesis de imágenes realistas.

Una innovación diferente a la interpretación de los Sistemas-L fue propuesto por Szilard y Quinton en 1979 [39], ellos concentraron la representación de imágenes con rigurosa definición geométrica (tales como códigos de cadenas) y mostraron que los Sistemas-L libres de contexto podrían generar intrínsecamente curvas llamadas hoy en día **fractales**; extendiendose posteriormente en diversas direcciones. Siromoney y Subramanian utilizaron a los Sistemas-L para generar las clásicas curvas de llenado [37]. Posteriormente Dekking investigó las propiedades limitantes de las curvas generadas por los Sistemas-L [12] y se concentró en la determinación de la dimensión fractal (Hausdorff). Prusinkiewicz presentó más ejemplos de fractales y estructuras de plantas usando Sistemas-L.

3.2. CARACTERISTICAS DE LOS SISTEMAS-L

La diferencia esencial entre las gramáticas de Chomsky y los Sistemas-L estriba en la manera de aplicar las producciones, en las gramáticas de Chomsky la aplicación se realiza secuencialmente, es decir una a la vez; mientras que, en los Sistemas-L las producciones son aplicadas en paralelo y simultáneamente reemplazando todas las letras de una palabra dada.

Los Sistemas-L tienen **dos subclases** diferentes [35]:

1.- Los **Sistemas-0L**, en el cual los caracteres son reemplazados por cadenas.

2.- Los **Sistemas-IL**, en donde las reglas de sustitución son más complicadas.

El Sistema-L que usaremos será del primer tipo; los Sistemas 0L y IL denotan el generador de la clase de lenguajes libres de contexto y el generador de la clase sensitiva al contexto respectivamente en los Sistemas-L (fig. 3.2)

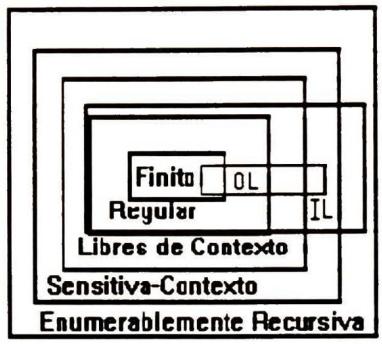


Fig. 3.2 Relación entre clases de lenguajes de Chomsky y las clases de lenguajes generados por los Sistemas-L.

A continuación mostraremos el siguiente recuadro (Rec 3.1) en donde se muestran 2 de los 3 tipos que tienen los Sistemas-L, a manera de ilustración para poder entender las diferencias que hay entre los Sistemas-L.

SISTEMAS-0L		SISTEMAS-2L	
N	L(N)	N	L(N)
0	1	0	1
1	0	1	0
2	01	2	11
3	010	3	00
4	01001	4	01(1)
5	01001010	5	111(0)
.....		.....	

Rec. 3.1

3.3. SISTEMAS-0L

El caso más simple de los Sistemas-L son los llamados Sistemas-0L. Empezaremos con un ejemplo el cual trata de demostrar de manera intuitiva el contenido de la idea principal de los Sistemas-0L. Considere las cadenas construidas por las letras **(ab)***; con las siguientes reglas de reescritura **a** -> **ab** y **b**-> **a**; el proceso de reescritura empieza con una cadena llamada axioma, que este ejemplo será **b**. En el primer paso de la derivación el axioma **b** es reemplazado por **a** usando la producción **b** -> **a**; en el segundo paso **a** es reemplazada por **ab** usando la regla de producción **a** -> **ab**; la palabra **ab** consiste de dos letras, ambas son simultáneamente reemplazadas en el siguiente paso de la derivación, de esta manera **a** es reemplazada por **ab** y **b** es reemplazado por **a** y la cadena resultante es **aba**; de una manera similar la cadena **aba** se extenderá a **abaab** y así sucesivamente (fig. 3.3).

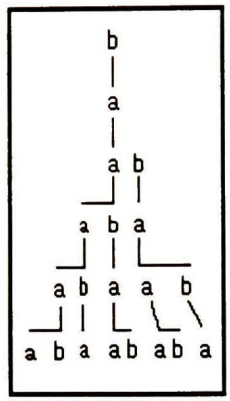


Fig. 3.3 Ejemplo de una derivación en los Sistemas-0L.

A continuación daremos una definición formal de los Sistemas-0L [35], un Sistema-0L es una tripleta ordenada G = <V, w, P> donde

V es el alfabeto del sistema.

$w \in V^+$ es una palabra no vacía llamada axioma o iniciador.

$P \subset V \times V^*$ es un conjunto finito de producciones.

el par (a,X) denota una producción "(a-->X)", en donde la letra a es el predecesor y la palabra X es el sucesor de esta producción. Se supone que para cualquier letra $a \in V$, hay al menos una palabra $X \in V^*$ tal que a --> X. Si una producción no es especificada explícitamente para un predecesor dado $a \in V$, se dice que la producción identidad a -> a pertenece al conjunto de producciones P.

Un Sistema-0L es determinístico si y solo si para cada $a \in V$, hay exactamente una $X \in V^*$ tal que a -> X.

A continuación ilustraremos la operación de los Sistemas-0L. Este ejemplo es usado para simular el desarrollo de un filamento multicelular encontrado en una bacteria Azul-Gris Anabaena Catenula y en varias algas. Los símbolos a y b representan estados citopatológicos de las células; los subíndices l y r indican la polaridad de la célula, especificando la posición en el cual los hijos de la célula de tipo a y b podrán ser reproducidos, el desarrollo es gobernado por las siguientes reglas:

$p_1 : a_r \rightarrow a_l\, b_r$

$p_2 : a_l \rightarrow b_l\, a_r$

$p_3 : b_r \rightarrow a_r$

$p_4 : b_l \rightarrow a_l$

A partir de una simple célula a_r (el axioma), el sistema de arriba genera la siguiente secuencia de palabras :

a_r

$a_l b_r$

$b_l\, a_r\, a_r$

$a_l\, a_l\, b_r\, a_l\, b_r$

el esquema correspondiente de la imagen generada por este ejemplo es mostrado en la fig. 3.4.

Una nota importante que cabe hacer mención de los Sistemas-0L, es que algunos Sistemas-0L tienen ciertas propiedades matemáticas, por ejemplo el de la figura 3.4, esta relacionado con los números de Fibonacci, ya que la longitud de las palabras generadas por este sistema corresponde casualmente a la serie de los números de Fibonacci.

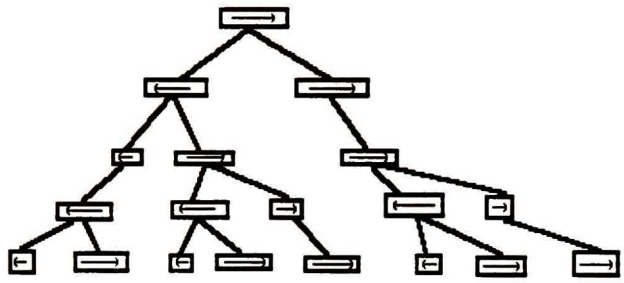


Fig 3.4. Desarrollo de un filamento (Anabaena Catenula) usando Sistemas-0L

3.4. MODELOS GRAFICOS DE LOS SISTEMAS-0L

3.4.1 CURVAS FRACTALES

Muchos fractales (o al menos aproximaciones finitas) se pueden pensar como una secuencia de elementos primitivos (líneas), la generación de cadenas por los Sistemas-0L deberá de contener la información necesaria acerca de la figura geométrica; la interpretación gráfica que se siguió fue la basada en la notación de la tortuga de Logo (fig. 3.5), esta interpretación fue originalmente propuesta por Szilard y Quinton [39]. Para este ejemplo se tomo como ángulo de rotación 90^{o}.

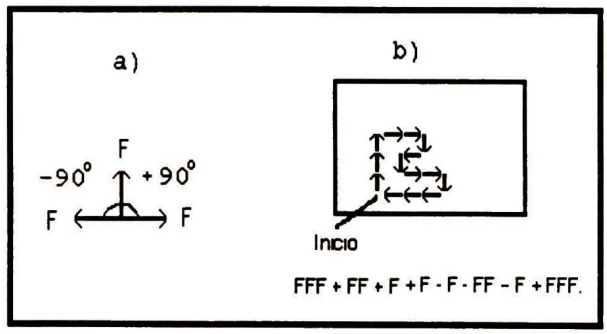


Fig 3.5 a) Interpretación de la Tortuga de los símbolos +,-
b) Interpretación de una gramática.

Un estado de la figura es definida con una tripleta (x,y,α) donde las coordenadas cartesianas (x,y) representan la posición de la tortuga y α el ángulo de rotación; dado el tamaño d y α, la tortuga puede responder a los siguientes comandos (fig. 3.6).

F.- Se mueve hacia adelante una longitud **d:** el estado de la tortuga (x, y, α) cambia a (x', y', α') donde x' = x + d cos(α) y y' = y + sin (α), un segmento entre los puntos (x , y) y (x' , y') es pintada
f.- Se mueve hacia adelante una longitud sin pintar una linea
+.- Gira a la derecha un ángulo (α)
- .- Gira a la izquierda un ángulo (α)

Fig. 3.6 Comandos que reconoce la Tortuga .

En el siguiente recuadro (Rec. 3.2) se muestra una pequeña parte del código para la generación de la curva de Koch utilizando Gramáticas de Lindermayer.

```
/* Gramáticas de Lindermayer  */
.....
Koch(Depth, Tam)
turn(90);  /* Gira 90 grados  */
Koch(Depth, Tam);
turn(90);
.....

void Koch(int Depth, int Tam)

if (Depth == 0 )
  ford(Tam);     /* Pinta una  Linea */
else
  Koch(Depth - 1, Tam);
  turn(90);
  Koch(Depth - 1, Tam);
  turn(90);
  Koch(Depth - 1, Tam):
   .....
```

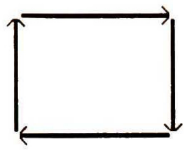
Rec. 3.2

EJEMPLOS:

A continuación daremos unos ejemplos de los Sistemas-0L para generar algunos fractales.

a) La Isla Cuádrica de Koch (fig 3.7)

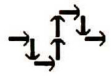
La isla cuádrica de Koch se obtuvo interpretando la cadena generada por el siguiente Sistema-0L:
Primero lo que deberemos de hacer es representar el iniciador de la isla cuádrica de Koch, como en este ejemplo el generador es un cuadrado y el ángulo de rotación es de 90° entonces tenemos:



F + F + F + F

Iniciador **Comandos**

Enseguida deberemos de representar el generador de la isla cuádrica de Koch por medio de los comandos de la tortuga de la siguiente manera:



F + F − F − FF + F + F − F

Generador **Comandos**

Por lo tanto como el iniciador es el axioma en el Sistema-0L y el generador es la regla de producción de este sistema se tiene que

w: F + F + F + F
p: F -> F + F - F - FF + F + F - F.
$\delta = 90^{o}$.

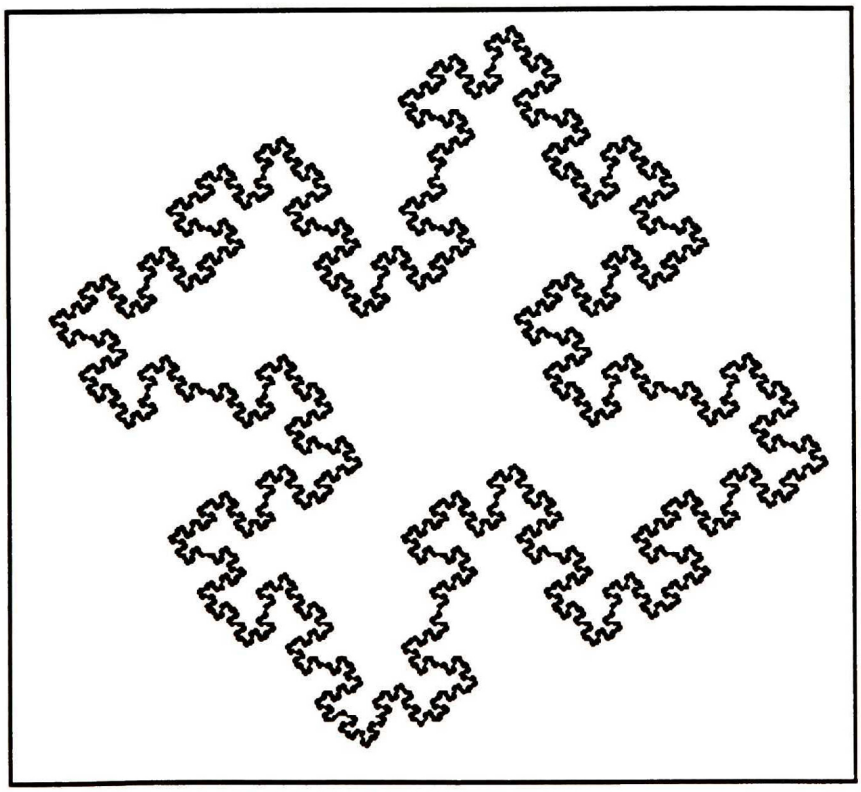


Fig 3.7 Isla Cuádrica de Koch

b) Curva de Copo de nieve (fig 3.8)

w: + F

p: F -> F - F ++ F - F.

$\delta = 60^o$

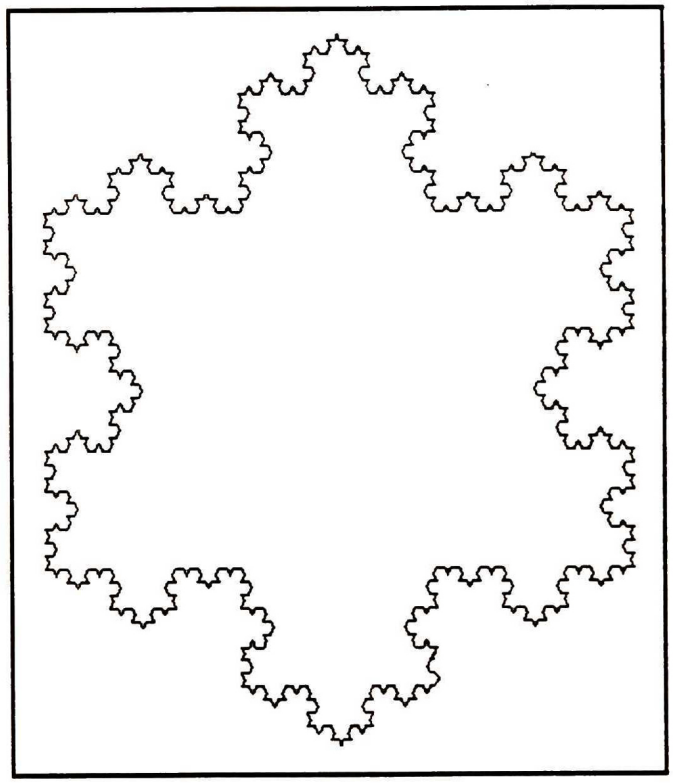


fig 3.8 Curva Copo de Nieve

c) Islas y lagos de Mandelbrot (fig 3.9).

w : F - F - F - F.

p : F -> F - f + FF - F - FF - Ff - FF + f - FF + F
+ FF + Ff + FFF.

f -> ffffff.

$\delta = 90^o$

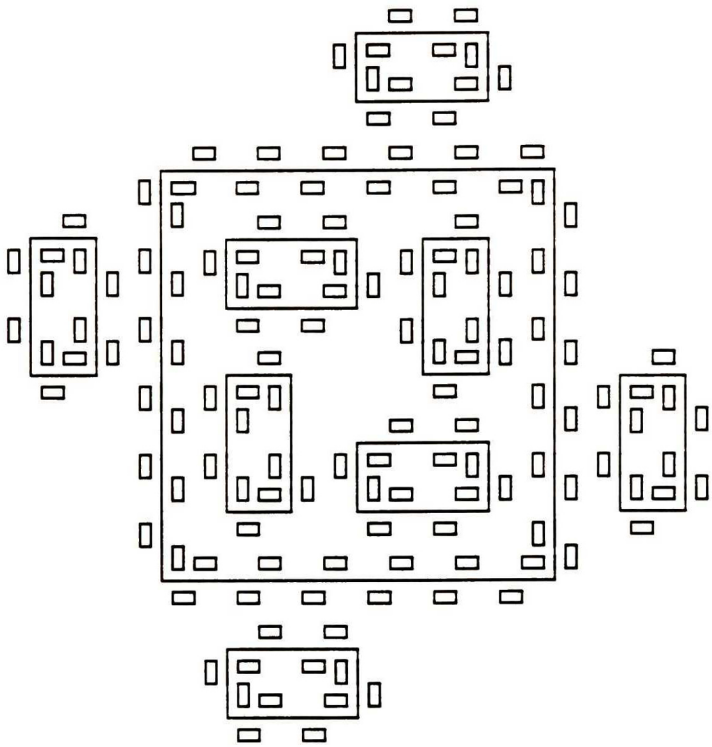


fig 3.9 Islas y Lagos de Mandelbrot

d) Cabeza de Flecha de Sierpinski (fig 3.10).

w : YF.

p : X --> YF + XF + Y.

Y --> XF - YF - X.

$\delta = 60^o$

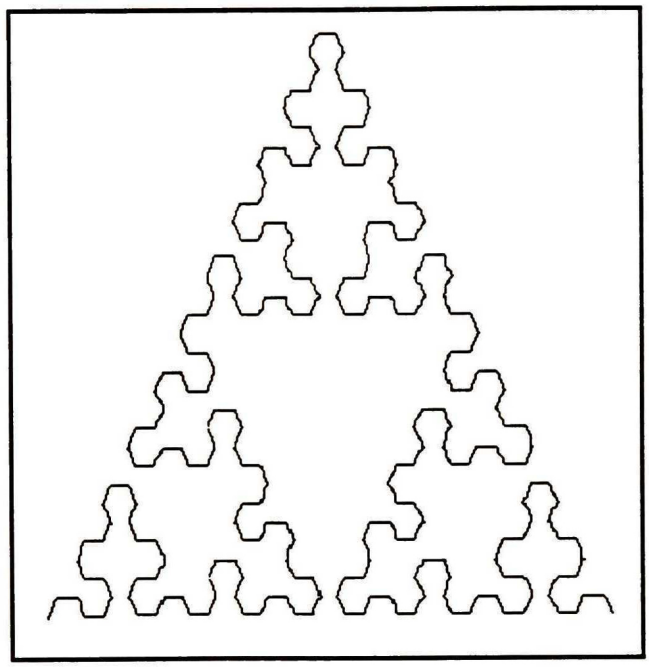


fig 3.10 Cabeza de flecha de Sierpinski

3.4.2 PLANTAS Y ARBOLES

En 1968 Lindermayer introdujo una notación para la representación gráfico-teórica de los árboles usando cadenas con corchetes [36]; la motivación fue formalizada describiendo estructuras bifurcadas encontradas en muchas plantas, que van desde las algas hasta los árboles, usando un esqueleto general de los Sistemas-L.

Posteriormente, la interpretación geométrica de los Sistemas-L que operaba en cadenas con corchetes fue introducida con el propósito de presentar estructuras modeladas e imágenes realistas; una extensión de la interpretación de la tortuga a las cadenas con corchetes y Sistemas-0L se describe a continuación:

Dos símbolos nuevos son interpretados por la tortuga:

[: Guarda el estado de la tortuga en una pila; la información salvada en la pila será la posición y orientación de la tortuga.
] : Se obtiene un estado de la pila y actualiza el estado de la tortuga; ninguna línea se traza, aunque la posición general de la tortuga cambia.

Un ejemplo de las cadenas con corchetes se muestra en la fig 3.11, donde el ángulo de rotación es de 45^{o}

F [-F] [+F [+F] F] F [-F] [+F]

Fig. 3.11 Interpretación de la tortuga, a cadenas con corchetes

A continuación se tiene una muestra de estas estructuras incluyendo su definición y producción gráfica.

EJEMPLOS:

Ejemplos de Sistemas-0L con corchetes para la modelación de plantas.

a) Espinoso (fig 3.12)

w F
p . F→ F[+F]F[-F]F
$\delta = 25^{\circ}$

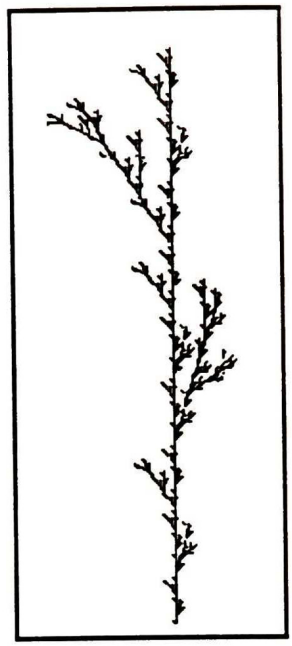


fig 3.12 Espinoso

b) Invernal (fig 3.13)

w : X
p : X→ F-[[X]+X]+
F[-FX]-X
F→ FF
$\delta = 22^{\circ}$

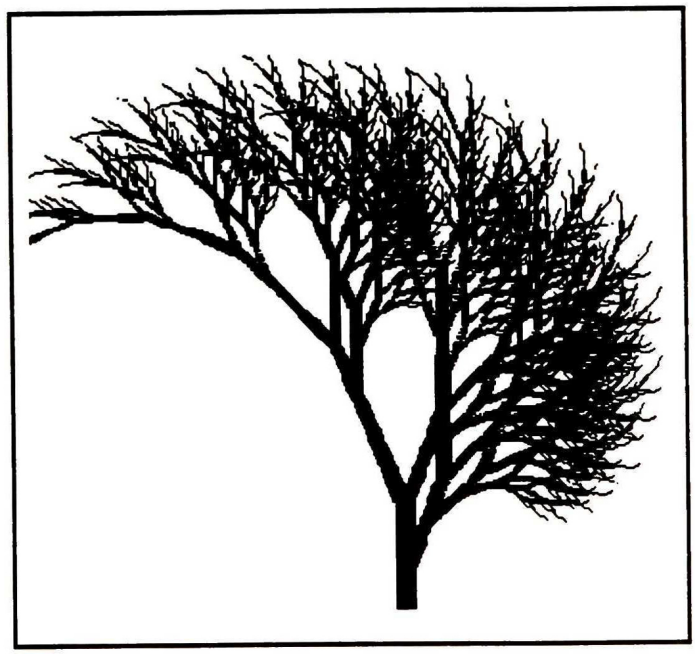


fig 3.13 Invernal

c) Arbol Aéreo (fig 3.14)

w : F
P : F → F F + [+ F - F - F] - [- F + F + F]
$\delta = 22^{\circ}$

fig 3.14 Arbol Aéreo

d) Abeto (fig 3.15)

w : S
p . S → [- - - G] [+ + + G] F S
G → - H [+ G] L
H → + G [- H] L
T → T L
L → [+ F F F] [- F F F] F
$\delta = 18^{\circ}$

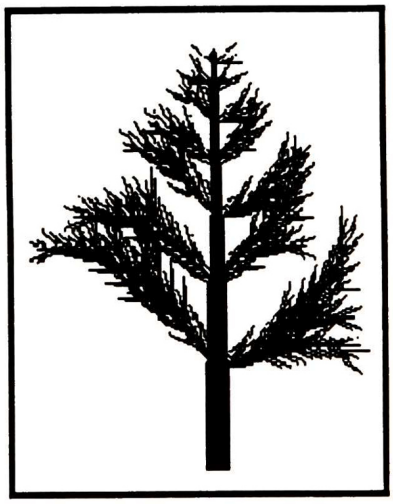


fig 3.15 Abeto

3.5 VENTAJAS Y DESVENTAJAS DE LOS SISTEMAS-0L

Las **ventajas** que nos proporcionan los Sistemas-0L son muy variadas:

1.- Son muy sencillas de comprender.
2.- Proporcionan una manera elegante para generar las clásicas curvas fractales.
3.- Fue diseñado especialmente para modelar topológicamente y geométricamente plantas, en particular los patrones de bifurcación de árboles y arbustos.
4.- El modelado de plantas se acerca mucho a la realidad.
5.- Es necesario proporcionar pocos datos para la creación de diferentes figuras (axioma y reglas de producción).
6.- No se requiere gran cantidad de memoria para almacenar datos.
7.- Se permite la elección de colores, para darle un mayor realismo a la figura.

Las **desventajas** que podemos enumerar son:

1.- Los árboles que se generan son simétricos.
2.- Todos los objetos modelados son fijos, no hay variaciones; dicho de otra forma, no se pueden obtener diferentes especímenes de la misma especie.

CAPITULO 4

TRIANGULACION Y MOVIMIENTO BROWNIANO

4.1. METODO DE TRIANGULACION

El principio sobre el cual se basa este método es el más sencillo de los cuatro métodos para generar fractales.

Por simplicidad se asume que el terreno que cubre es una área triangular; subdividimos el triángulo en cuatro pequeños triángulos por el desplazamiento aleatorio de medio punto de cada lado y juntamos los nuevos puntos con tres líneas rectas. Este desplazamiento aleatorio puede ser hacia arriba o hacia abajo, dependiendo de una variable aleatoria Gaussiana; por lo tanto cada triángulo puede ser dividido de la misma manera [14].

La cantidad de cada desplazamiento es determinada por una variable aleatoria Gaussiana con media 0 y varianza 1. Este proceso es descrito en la figura 4.1 y en el recuadro (Rec. 4.1) se muestra parte del código del método de Triangulación.

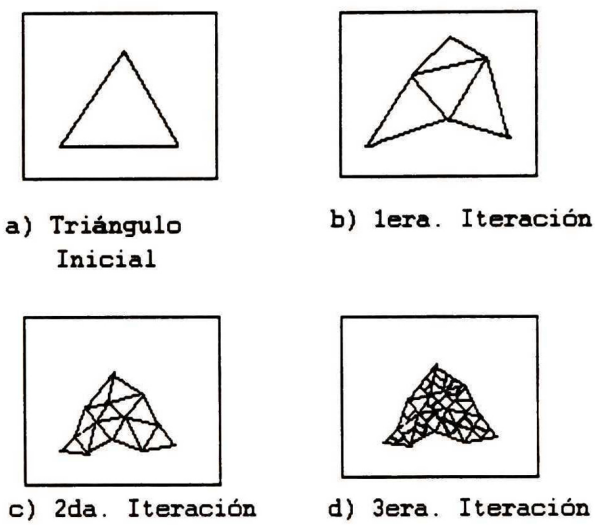


Fig. 4.1 Técnica de triangulación.

```
/* Método de Triangulación */
.....
Fh(0) = Gauss(seed)  x Scale: /* Obtiene el primer punto */
Fh(My) = Guass(seed) x Scale:
std = Scale * ratio:
Subdivide(0,My, std);
Moveg(Fh(0), 0);
for (i= 0; i <My ; i++)
   px1 = Fh(i)
   px2 = Fh(2);
.....
.....
Draw_L(px1, 2 x 1, px2 x (i+1), 3);
```

Rec. 4.1

MOVIMIENTO BROWNIANO

Una trayectoria puede ser descrita por una función $f:\mathbf{R}^+ \rightarrow \mathbf{R}^n$ donde f(t) es la posición de la partícula en un tiempo t. Podemos estudiar t de dos diferentes puntos de vista, cualquiera de estos puntos de vista puede ser pensado como una trayectoria o camino $f([t_1,t_2]) = \{f(t): t1 \leq t \leq t2\}$ como un subconjunto de $\mathbf{R}^n$ con t considerado como un parámetro o puede ser considerado como una gráfica de f, la gráfica $f = \{(t,f(t)): t1 \leq t \leq t2\}$, es como un registro de la variación de t con el tiempo.

Este estudio se iniciará con una investigación de la forma fractal del movimiento Browniano clásico, examinando algunas variantes que han sido usadas para modelar una amplia variedad de fenómenos.

4.2.1 CARACTERISTICAS DEL MOVIMIENTO BROWNIANO

El movimiento Browniano en una variable constituye el fractal aleatorio más simple y es la base para la definición en varias variables; el movimiento Browniano es también referenciado como "ruido Brown".

El movimiento Browniano en una dimensión lo definiremos de la siguiente manera: Considere una partícula recorriendo un camino aleatorio en una línea, suponiendo pequeños intervalos τ de saltos de la partícula a una pequeña distancia δ, aleatoriamente de izquierda a derecha . Sí $X_\tau(t)$ denota la posición de una partícula en un tiempo t, entonces damos la posición $X_\tau(k\tau)$ en el tiempo $k\tau$, $X\tau((k+1)+\delta)$ es semejante a $X_\tau(k\tau)+ \delta$ ó $X_\tau(k\tau)- \delta$, asumiendo que una partícula empieza en el origen en un tiempo 0, entonces para t>0, la posición en un tiempo t esta descrita por una variable aleatoria

$$X_\tau(t) = \delta(Y_1 + \ldots + Y_{[t/\tau]}) \qquad (4.1)$$

donde $Y_1, Y_2, ..$ son variables aleatorias independientes, en donde cada una tiene probabilidad 1/2 de tomar el valor 1 y probabilidad 1/2 de tomar el valor -1. Aquí $[t/\tau]$ denota el entero más grande menor o igual a t/τ; normalizando la longitud del paso como δ tenemos que

$$X_\tau(t) = \tau^{1/2} (Y_1 + ... + Y_{[t/\tau]}) \qquad (4.2)$$

El teorema del límite central nos dice que para una t fija, sí τ es el más pequeño (τ=0), entonces la distribución de la variable aleatoria $X_\tau(t)$ es aproximadamente normal con media 0 y varianza t, ya que las Y_i tienen media 0 y varianza 1; de la misma manera, si t,h son fijas y τ es suficientemente pequeña, entonces $X_\tau(t+h) - X_\tau(t)$ es aproximadamente normal con media 0 y varianza h, note que sí $0 \le t_1 \le t_2 ..... \le t_{2m}$, entonces los incrementos $X_\tau(t_2) - X_\tau(t_1)$, $X_\tau(t_4) - X_\tau(t_3)$,...,$X_\tau(t_{2m}) - X_\tau(t_{2m-1})$ son variables aleatorias independientes, definiendo el movimiento browniano como el límite del camino aleatorio $X_\tau(t)$ cuando $\tau \to 0$.

Definimos el movimiento Browniano o proceso de Wiener como un proceso aleatorio X tal que:

i) X(0) = 0 y X(t) es una función continua de t.
ii) Para cualquier $0 \le t$ y $h > 0$ el incremento X(t+h) - X(t) tiene una distribución normal con media 0 y varianza h, así

$$P(X(t+h) - X(t) \le x) = (2\pi h)^{-1/2} \int_{-\infty}^{x} \exp[-u^2 / 2h^2] du \qquad (4.3)$$

iii) si tenemos que $0 \le t_1 \le t_2 ..... \le t_{2m}$, los incrementos $X(t_2) - X(t_1)$,., $X(t_{2m}) - X(t_{2m-1})$ son independientes.

se sigue de i) y ii) que X(t) tiene una distribución Normal con media 0 y varianza t para cada t, observe que los incrementos de X son estáticos, esto es X(t+h) - X(t) tienen una distribución independiente de t.

El movimiento browniano puede ser construido por medio de los siguientes métodos [32]:

1.- Usando aproximaciones aleatorias (4.1), los valores de 1 o -1 son asignados por "el lanzamiento de una moneda" hacia Y_i para $1 \le i \le m$, donde m es muy grande y $X_\tau(t)$ es pintado adecuadamente; sí τ es pequeño comparado con t, entonces este podrá dar una buena aproximación a una simple función Browniana.

2.- El otro método es "El desplazamiento aleatorio del medio punto", se empieza X(0) = 0 y seleccionando X(1) como una variable aleatoria Gaussiana con media 0 y varianza σ^2. Entonces $var(X(1) - X(0)) = \sigma^2$ y se espera

$$var(X(t_2) - X(t_1)) = | t_2 - t_1 | \sigma^2 \qquad (4.4)$$

para $0 \leq t_1 < t_2 \leq 1$, poniendo $X(1/2)$ como el promedio de $X(0)$ y $X(1)$ más algún desplazamiento aleatorio Gaussiano D_1 con media 0 y varianza Δ_1^2, entonces

$$X(1/2) - X(0) = 1/2\,(X(1) - X(0)) + D_1$$

y así $X(1/2) - X(0)$ tienen media 0 y lo mismo pasa para $X(1) - X(1/2)$.

Siguiendo con el método se tiene por la ecuación (4.4)

$$\text{var}(X(1/2) - X(0)) = 1/2\ \text{var}(X(1) - X(0)) + \Delta_1^2 = 1/2\sigma^2$$

por lo tanto

$$\Delta_1^2 = 1/4\sigma^2$$

en el siguiente paso se procede de la misma forma

$$X(1/4) - X(0) = 1/2\,(X(0) + X(1/2)) + D_2$$

y se observa los incrementos en X, aquí $X(1/2)$-$X(1/4)$ y $X(1/4)$-$X(0)$ son Gaussianas con media 0. Por lo tanto la varianza de Δ_2^2 de D_2 es

$$\text{var}(X(1/4) - X(0)) = 1/4\ \text{var}(X(1/2) - X(0)) + \Delta_2^2 = 1/4\sigma 2$$

es decir

$$\Delta_2^2 = 1/8\ \sigma^2$$

Continuando con la misma idea se tiene entonces que en la n-esima etapa se tendrá:

$$\Delta_n^2 = (1/(2n+1))\ \sigma^2 \qquad (4.5)$$

como la varianza del desplazamiento D_n.

La gráfica de una función Browniana es mostrada en la fig. 4.2

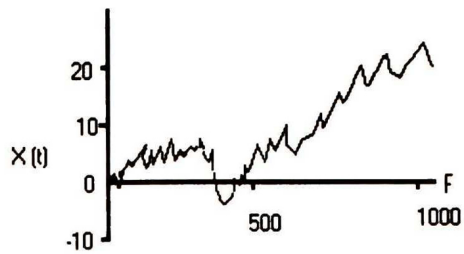


Fig. 4.2 Gráfica de una función Browniana.

Es fácil extender la definición del movimiento Browniano de $\mathbf{R}$ a $\mathbf{R}^n$, definiendo el movimiento Browniano en $\mathbf{R}^n$ de modo que los componentes de las coordenadas son movimientos independientes Brownianos en una dimensión. Así $X:[0, \infty) \rightarrow \mathbf{R}^n$ dados por $X(t)=(X_1(t),..,X_n(t))$ en un movimiento Browniano de n dimensiones en algún espacio de probabilidad si los procesos aleatorios $X_i(t)$ están en un movimiento Browniano de una dimensión para cada i, y $X_1(t_1),..,X_n(t_n)$ son independientes para cualquier conjunto de tiempos $t_1,...t_n$; una trayectoria de movimientos Brownianos en $\mathbf{R}^2$ es mostrado en la fig. 4.3 y en el recuadro 4.2 se muestra parte del programa que genera el movimiento Browniano en dos dimensiones.

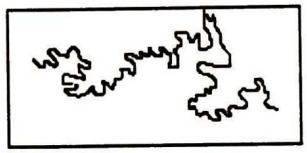


Fig. 4.3 Una trayectoria Browniana en $\mathbf{R}^2$

La proyección de X(t) en cada uno de los ejes es un movimiento Browniano de una dimensión, ademas los ejes no son especiales a este respecto ya que un movimiento Browniano de n dimensiones es isotrópico (esto significa que tienen las mismas características en todas las direcciones). Para checar esto considere por conveniencia el caso del movimiento Browniano en dos dimensiones $X(t)=(X_1(t), X_2(t))$, la proyección de X(t) sobre una línea $L\phi$ en un ángulo ϕ que pasa por el origen es $X_1(t)\cos\phi + X_2(t)\sin\phi$. Para $t > 0$ y $h > 0$ las variables aleatorias $X_1(t+h) - X_1(t)$ y $X_2(t+h) - X_2(t)$ son independientes y con una distribución normal con media 0 y varianza h, así los incrementos de la proyección en L_ϕ están dados por:

$$(X_1(t+h) - X_1(t))\cos\phi + (X_2(t+h) - X_2(t))\sin\phi \qquad (4.6)$$

donde son normalmente distribuidos con media 0 y varianza $h\cos^2\phi+h\sin^2\phi=h$; de una manera similar, los incrementos de la proyección son independientes, así la proyección X(t) en L_ϕ es un movimiento Browniano de una dimensión, para todos los ángulos ϕ.

```
/* Algoritmo para crear el Movimiento Browniano en dos dimensiones */

void Browniana()
{
   float scale=1000,h=.87,std;
   int parte, My=300;

   scanf("%d",&parte);
   Fh[0] = gauss(parte) * scale;
   Fh[My] = gauss(0) * scale;
   ratio = pow(2,-h);
   std = scale*ratio;
   subdivide(0,My,std);
}

void subdivide (int f1, int f2, float std)
{
  int fmid;
  float stdmid;

  fmid = (f1 + f2)/2;
  if (( fmid != f1) && (fmid != f2))
  {
    Fh[fmid] = (Fh[f1] + Fh[f2])/2.0 + gauss(0) * std;
    stdmid = std*ratio;
    subdivide(f1,fmid,stdmid);
    subdivide(fmid,f2,stdmid);
  }
}
```

Rec 4.2

4.3 MOVIMIENTO BROWNIANO FRACCIONAL

Uno de los más útiles modelos matemáticos para fractales aleatorios encontrados en la naturaleza (tales como montañas y nubes) ha sido el movimiento Browniano fraccional (fBm) de Mandelbrot y Van Ness ([28], [29]), esta es una extensión del movimiento Browniano. Casi todas las simulaciones de la naturaleza con fractales se basan en la extensión del fBm para dimensiones altas [40].

El movimiento Browniano fraccional es un buen punto de partida para el entendimiento de difusiones análogas y caminos aleatorios en fractales. Unas trayectorias simples de fBm son mostradas en la fig. 4.4, VH(Tt) es una simple función dependiente de una sola variable t(usualmente el tiempo). En apariencia el movimiento es muy parecido a las montañas lejanas o a una fluctuación de una economía variable.

El comportamiento del escalamiento de las diferentes trayectorias de la fig. 4.4 se caracteriza por el parámetro H el cual se encuentra entre 0<H<1; cuando H es cercano a 0 las trayectorias son muy ásperas mientras que cuando H tiende a 1 las trayectorias son relativamente más suaves. H relaciona los cambios típicos en V, $\Delta V=V(t_2)$ $V(t_1)$, para la diferencia del tiempo $\Delta t=t_2 - t_1$ por una ley simple de escalamiento:

$$\Delta V \alpha \Delta t^h \qquad (4.8)$$

en el movimiento Browniano usual, la suma de los incrementos o pasos independientes nos llevan a una variación de escalas como las raíces cuadradas de los pasos, así h=1/2 corresponde a la trayectoria del movimiento Browniano.

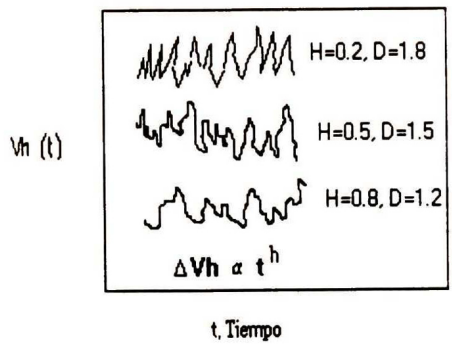


Fig. 4.4 Ejemplos de Movimientos Brownianos Fraccionales

Un movimiento Browniano fraccional VH(t), es una función simplemente valuada de una variable t (usualmente el tiempo), estos incrementos $V_H(t_2) - V_H(t_1)$ tienen una distribución Gaussiana con varianza

$$[< | V_H(t_2) - V_H(t_1) | >^2] \alpha |t_2 - t_1|^{2H} \qquad (4.9)$$

donde los '< >' denotan los porcentajes sobre muchas muestras $V_H(t)$ y el parámetro H tiene un valor entre $0 < H < 1$, tal función es fija e isotropa; estos incrementos de las medias cuadráticas dependen solamente de la diferencia del tiempo $t_2 - t_1$ y todos los t's son estadísticamente equivalentes; el valor especial H=1/2 da un movimiento Browniano con

$$\Delta V^2 \alpha \Delta t \qquad (4.10)$$

de esta manera el movimiento Browniano; aunque $V_H(t)$ es continuo no es diferenciable en ninguna parte, la derivación del movimiento Browniano normal, H=1/2 corresponde a una Gaussiana no correlacionada (fig 4.5a)

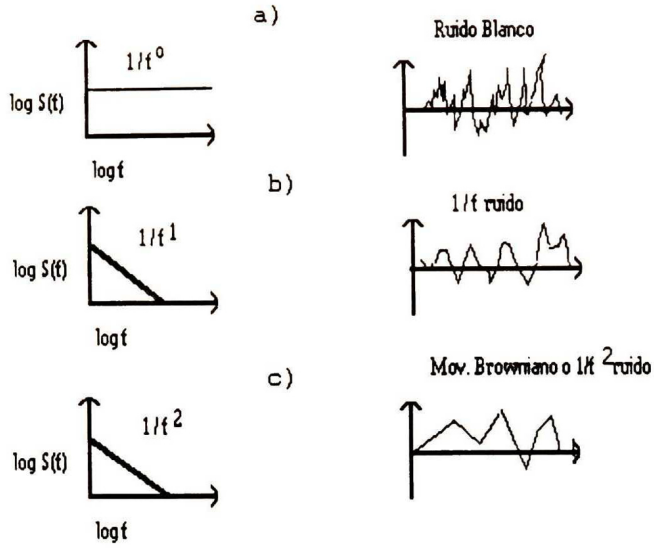


Fig. 4.5 Muestras de ruidos típicos, V(t), las variaciones aleatorias en una cantidad de tiempo.
a) Ruido Blanco b) 1/f ruido. c) Movimiento Browniano.

Para H > 1/2 hay una correlación positiva para los incrementos de $V_H(t)$ y esta es la derivada del ruido fraccional Gaussiano, para H < 1/2 los incrementos se correlacionan negativamente, tales correlaciones se extienden para una longitud arbitraria de tiempos escalados y tienen un efecto de apariencia visual de las trayectorias fBm (fig. 4.4.)

Resumiendo el movimiento Browniano fraccional de índice $\alpha(0<\alpha<1)$ es definido como un proceso aleatorio $X:[0,\infty)\rightarrow\mathbf{R}$ en algún espacio de probabilidad tal que:

i) Con probabilidad 1, X(t) es continuo y X(0)= 0.

ii) Para cualquier t > 0 y h > 0 los incrementos X(t+h) - X(t) tienen una distribución normal con media 0 y varianza $h^{2\alpha}$, así que:

$$P(X(t+h) - X(t) \leq x) = (2\pi)^{-1/2} h^{-\alpha} \int_{-\infty}^{x} \exp(-u^2/2h^{2\alpha})\, du \qquad (4.11)$$

como se puede ver en la definición de arriba los incrementos X(t+h) - X(t) son estáticos. No obstante las funciones de distribución especificadas por fBm no pueden tener incrementos independientes excepto en el caso cuando $\alpha=1/2$, por la condición (ii) $E((X(t+h) - X(t))^2) = h^{2\alpha}$, para lo cual puede ser mostrado que

$$E(X(t)\,(X(t+h) - X(t))) = 1/2[(t+h)^{2\alpha} - t^{2\alpha} - h^{2\alpha}] \qquad (4.12)$$

el cual es diferente de cero si $\neq 1/2$.

CAPITULO 5

FRACTALES TRIDIMENSIONALES

5.1.- SUPERFICIES BROWNIANAS

Como se explicó anteriormente en el capítulo 4, el movimiento Browniano puede ser extendido a 3 dimensiones para generar Superficies Brownianas [36], esta ampliación es bastante simple, ya que tomaremos como base el movimiento Browniano bidimensional.

El movimiento Browniano bidimensional fue realizado de la siguiente manera:

$$(X_i , X_i(t)) \text{ ó } (X_i(t) , X_i) \qquad (5.1)$$

en donde X_i es el desplazamiento de la gráfica y $X_i(t)$ es una variable Gaussiana, de esta manera se obtiene una pareja de puntos que pintan para formar la gráfica del movimiento Browniano bidimensional.

Continuando con esta idea, extenderemos el movimiento Browniano de 2 dimensiones a 3 dimensiones, para poder realizar esto necesitaremos agregar una variable mas a (5.1), la cual será la altura de la Superficie Browniana en los puntos x,y; por lo tanto las superficies Brownianas son especificadas como:

$$(X_1 , X_2 , X_i(t)) \qquad (5.2)$$

donde X_1 y X_2 es el desplazamiento (x,y) que se dará a la Superficie Browniana y X_i es una variable Gaussiana. La triada de puntos que iremos obteniendo serán almacenados en un archivo, el cual posteriormente será usado por el algoritmo que se encuentra en la sección 3.1 de este capítulo, para graficar la Superficie Browniana que se generó.

EJEMPLOS

A continuación daremos dos ejemplos de Superficies Brownianas.

a) Superficie Browniana 1 (fig 5.1)

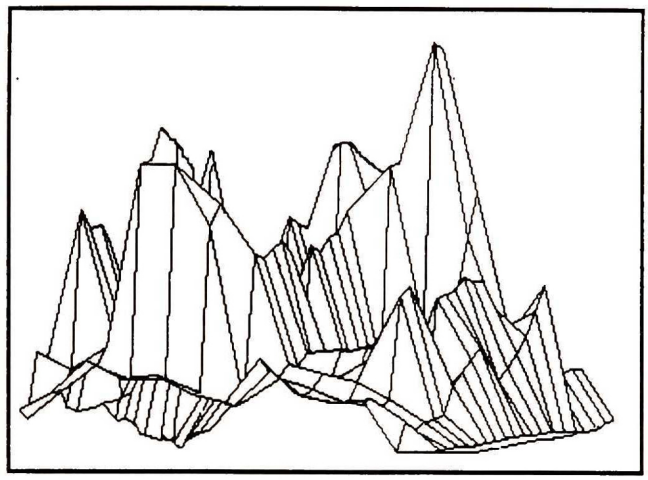


Fig 5.1

b) Superficie Browniana 2 (fig. 5.2)

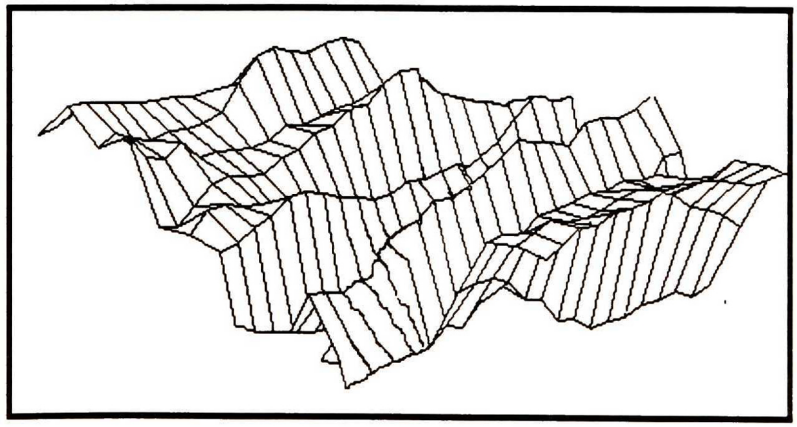


Fig 5.2

5.2. SISTEMAS-0L

5.2.1.- CURVAS FRACTALES

La generación de las curvas fractales tridimensionales será una extensión de las curvas fractales bidimensionales que fueron explicadas anteriormente en el capítulo 3.4.1; para poder realizar las curvas fractales tridimensionales [36], necesitaremos introducir nuevos comandos que deberán ser interpretados por la tortuga (fig. 5.3) los cuales son los siguientes:

+.- Gira a la izquierda un ángulo δ el eje Z. La matriz de rotación es igual a RZ(δ).

-.- Gira a la derecha un ángulo δ el eje Z. La matriz de rotación es igual a RZ(-δ).

&.- Gira a la derecha un ángulo δ el eje Y. La matriz de rotación es igual a RY(δ).

^.- Gira a la izquierda un ángulo δ el eje Y. La matriz de rotación es igual a RY(-δ).

\.- Gira a la derecha un ángulo δ el eje X. La matriz de rotación es igual a RX(δ).

/.- Gira a la izquierda un ángulo δ el eje X. La matriz de rotación es igual a RX(-δ).

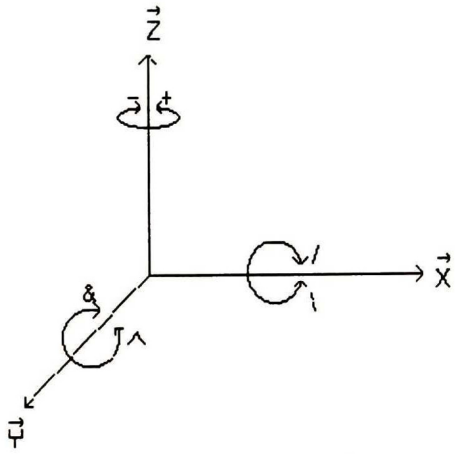


Fig 5.3 Rotación de los ejes de acuerdo a la tortuga.

en donde las matrices de rotación RZ, RY y RX son:

$$RZ(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$RX(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

$$RY(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

una vez introducidos estos comandos, se podran construir las curvas fractales tridimensionales. La clave de este método es representar la orientación de la tortuga en el espacio tridimensional por medio de las matrices de rotación RX, RY y RZ.

EJEMPLOS

A continuación daremos un ejemplo ilustrativo de como generar una curva fractal tridimensional. El ejemplo que tomaremos como referencia será el de la isla cuádrica de Koch, como se recordará en la sección 3.4.1 se dió una especificación de la isla cuádrica de Koch bidimensional de la cual se dedujo lo siguiente:

w: F + F + F + F
p: F --> F + F - F - F F + F + F - F.
$\delta = 90^o$.

ahora para generar la isla cuádrica de Koch tridimensional, deberemos ir generando planos (en lugar de líneas) e ir trasladando el origen de las coordenadas a los puntos que se vayan obteniendo; así que el axioma y las reglas de producción de la isla cuádrica de Koch son:

w: & - XS - XS - XS - XS
p: X -> XS + XS -XS - XSXS + XS + XS - X
S -> F ^ F ^ F ^ F ^ F
$\delta = 90^o$.

Una vez que tenemos estos tres elementos (axioma, regla de producción y ángulo), para poder graficar la isla cuádrica de Koch, deberemos aplicar el axioma y la regla de producción (con el nivel de recursión especificado), e ir paralelamente almacenando en un archivo los puntos (x,y,z) que se van generando y emplear el algoritmo especificado en la Sección 3.1. de este capítulo.

La figura resultante que se obtuvo al aplicar estos pasos es mostrada en la figura 5.4.

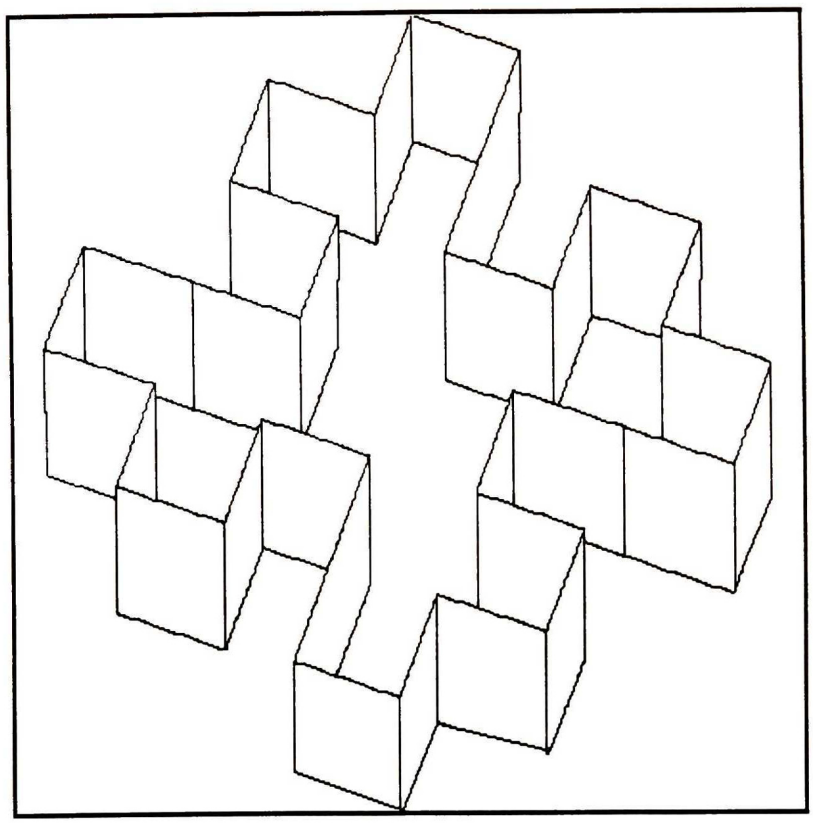


Fig 5.4 Isla Cuádrica de Koch Tridimensional

A continuación daremos otros dos ejemplos de curvas fractales tridimensionales.

a) Curva de Hilbert (fig 5.5)

w : X
X -> -YS + XSX + SY
Y -> +XS - YSY - SX +
S -> F ^ F ^ F ^ F^ F.
δ : 90º

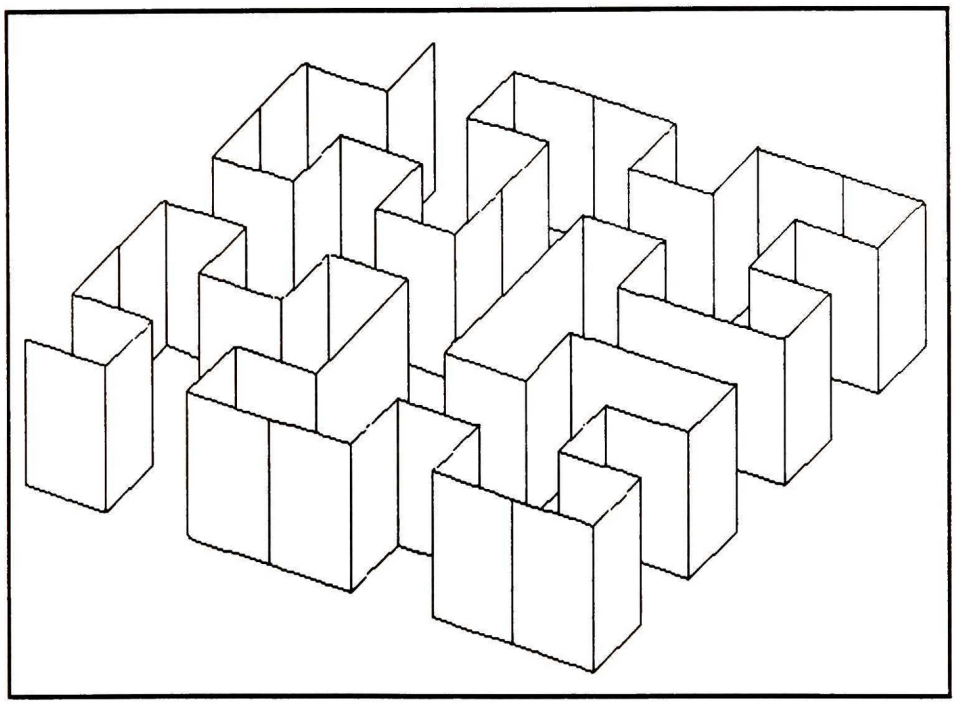


Fig 5.5 Curva de Hilbert Tridimensional

b) Curva de Koch Cuádruple (fig 5.6)

w : & + XS + XS + XS + XS

X -> XSXS + XS + XS + XS + XSX

S -> F ^ F ^ F ^ F ^ F

δ : 90º

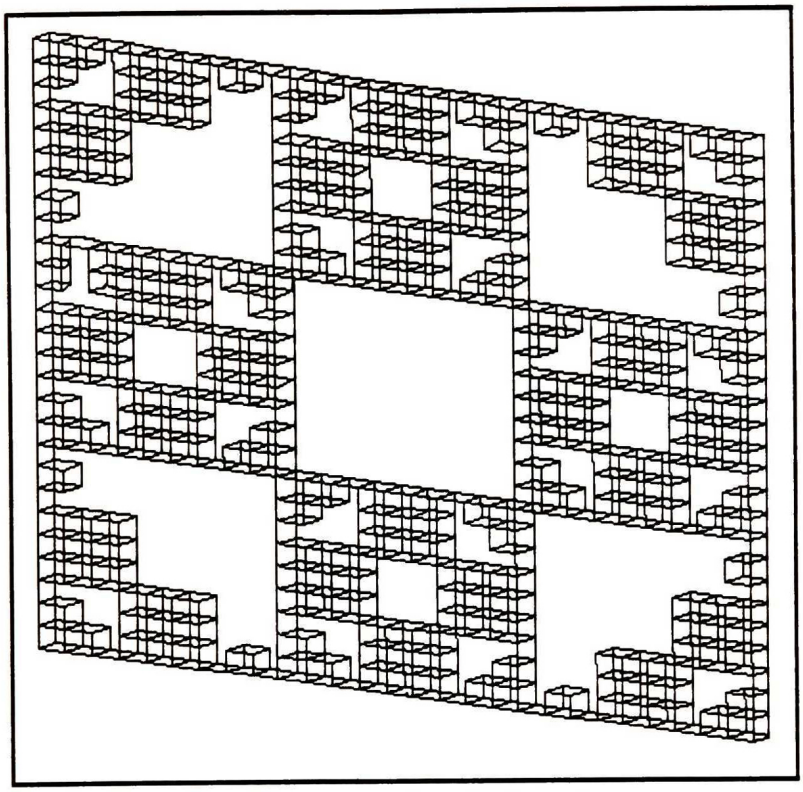


Fig 5.6 Curva de Koch Cuádruple Tridimensional

5.2.2. PLANTAS

La generación de las plantas tridimensionales tiene sus bases en las plantas y árboles bidimensionales explicadas anteriormente en el capítulo 3.4.2. El proceso de generación de las plantas tridimensionales es muy similar a las plantas bidimensionales, ya que la principal diferencia que se tiene entre estas, es que los elementos terminales no van hacer líneas si no que ahora van hacer superficies tales como (hojas, ramas, etc.), por lo tanto podremos generar figuras tridimensionales tales como rosas, orquídeas, etc.

El proceso de generación de cualquier planta tridimensional consiste de tres pasos fundamentalmente y son:

Primero.- Definir el conjunto de reglas: axioma y reglas de producción que deberá de tener la planta.

Segundo.- Definir los puntos que formaran los elementos terminales de nuestra planta; para esto el proceso que se utilizó fue el de dibujar el elemento terminal en una hoja milimétrica y determinar los puntos que forman a este objeto.

Tercero.- Aplicar el conjunto de reglas para obtener a la planta deseada.

La graficación de plantas tridimensionales consiste básicamente en ir almacenando en un archivo los puntos que se van generando al aplicar el conjunto de reglas y después ejecutar los algoritmos especificados en las secciones 3.1 y 3.2 respectivamente para obtener la figura deseada.

EJEMPLOS

A continuación daremos dos ejemplos de plantas tridimensionales:

1.- El primer ejemplo que daremos será el de una rosa el cual tuvo como resultado el siguiente conjunto de reglas :

a) Rosa (fig 5.7)
w : Rosa
Rosa -> Tallo + Pétalo + Pétalo + Pétalo+ Pétalo
Tallo -> FFF + F + F + F.
Pétalo -> Es la superficie obtenida en la hoja milimétrica.

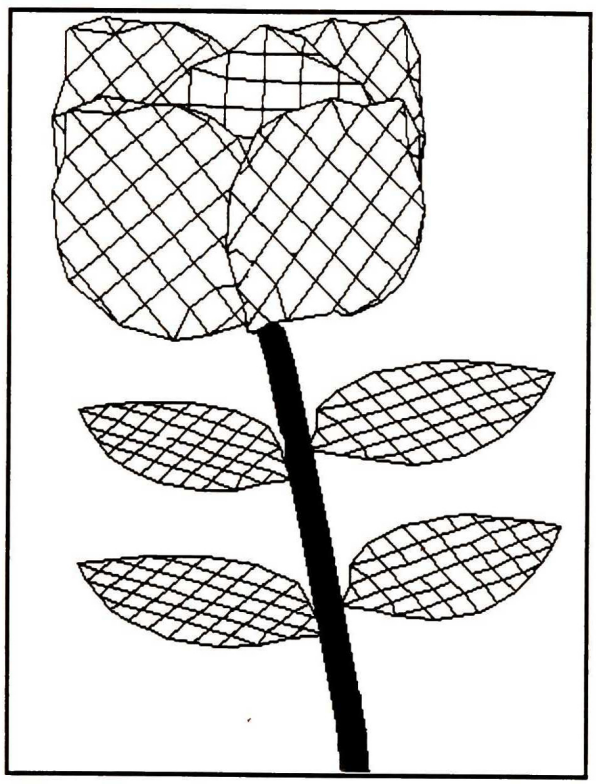


Fig 5.7 Rosa

El segundo ejemplo que se consideró fue el de una Orquídea el cual arrojó el siguiente conjunto de reglas :

b) Orquídea (fig 5.8)

w: Orquídea

Orquídea -> Tallo + Hoja + Hoja + Hoja + Hoja + Hoja + Hoja

Tallo -> FF + F + FF.

Hoja -> Es la superficie de la hoja que obtuvo en la hoja milimétrica.

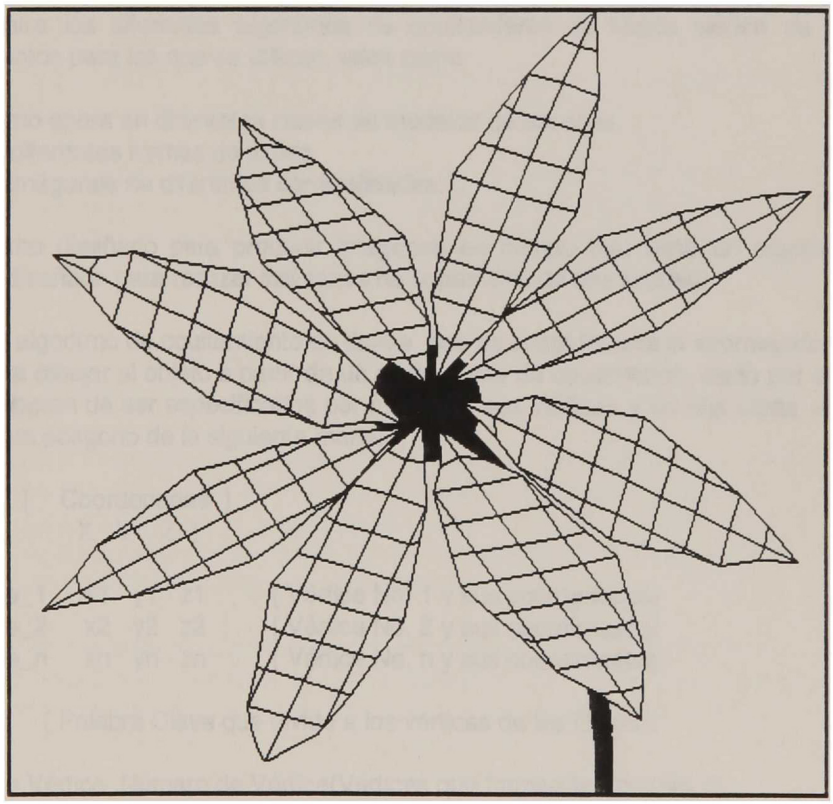


Fig 5.8 Orquídea

5.3. ALGORITMOS GENERALES

5.3.1.- CARACTERISTICAS DEL ALGORITMO DE OCULTAMIENTO DE LINEAS.

De todas las áreas que se tienen en graficación por computadora ninguna de ellas ha recibido tanta atención como el problema denominado "Ocultamiento de Líneas". El problema del ocultamiento de líneas es determinar cuales aristas de un sólido son visibles y cuales no son visibles vistas desde un punto de observación.

A principios de los 60's, Robert realizó el primer algoritmo de ocultamiento de líneas, el cual resulto ser bastante lento para dibujar objetos muy simples; posteriormente al trabajo de Robert, se inició una carrera por desarrollar un algoritmo mas rápido a este. Dentro de los trabajos que podemos mencionar por sus méritos están el de Ruth Weiss, (Warnock, Watkins y Newell), Sproull y Sutherland y Hodgman.

Con la existencia de infinidad de algoritmos de ocultamiento de líneas ¿ Cual de ellos es el mejor ?, la respuesta a esta pregunta quizá nunca tenga respuesta, ya que las diferencias que existen entre los diferentes algoritmos de ocultamiento de líneas vienen de los diferentes requerimientos para los que se utilizan, tales como:

- el algoritmo opera en diferentes clases de modelos de escenas.
- generan diferentes formas de salida.
- produce imágenes de diferentes complejidades.

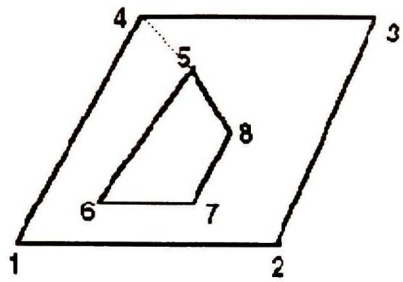
Un algoritmo diseñado para producir imágenes en tiempo real tiene un objetivo diferente al algoritmo diseñado para realizar imágenes de superficies de alta calidad.

El algoritmo de ocultamiento de líneas, que se utilizó tomará la información de un archivo ascc'ii, para dibujar al objeto a partir de un cierto punto de observación, dado por el usuario. Los objetos deberán de ser especificados por medio de sus vértices y de sus caras, en donde cada cara será un polígono de la siguiente manera:

```
            {  Coordenadas }
{    i          X  Y   Z }

No_Vértice_1    x1  y1  z1      { Vértice No. 1 y sus coordenadas}
No_Vértice_2    x2  y2  z2      { Vértice No. 2 y sus coordenadas}
No_Vértice_n    xn  yn  zn      { Vértice No. n y sus coordenadas}

Caras:      { Palabra Clave que divide a los vértices de las Caras }

Número de Vértice  Número de Vértice{Vértices que forman la Cara No.1}
Número de Vértice  Número de Vértice{Vértices que forman la Cara No.2}
.....
.....
Número de Vértice  Número de Vértice{Vértices que forman la Cara No.n}
```

Esta es la estructura que deberá de guardar el archivo para poder utilizar el algoritmo de ocultamiento de líneas. Las palabras que se encuentran entre paréntesis solo son comentarios y estos no deberán aparecer en el archivo; la estructura de este archivo impone al algoritmo de ocultamiento de líneas las siguientes restricciones:

1.- Los números de los vértices que formaran las caras deberán de ser especificadas en sentido contrario a las manecillas del reloj con respecto al punto de vista del observador.
2.- Si una cara tiene agujeros se introducirán aristas artificiales (fig. 5.9)
3.- Los primeros 3 vértices de una cara deberán de formar un ángulo convexo.



1 2 3 4 -5 8 7 6

Fig. 5.9 Arista Artificial

Antes de utilizar el algoritmo de ocultamiento de líneas, explicaremos brevemente los pasos previos que se realizan para utilizar dicho algoritmo.

Ya que se tiene el nombre del archivo en donde se encuentra la información del objeto a dibujar; los vértices del objeto se irán almacenando en el arreglo VERTEX, finalizado este paso se procederá a leer las caras que forman a dicho objeto, cada cara que sea leida se irá dividiendo en triángulos y estos serán almacenados en el registro TRIANGLE el cual tienen los siguientes campos:

a1, b1, c1 : Número de los vértices que forman el triángulo.
a,b,c,h : Coeficientes de la Ecuación del plano que pasa por ese triángulo.

Un paso importante que deberemos de tener en cuenta antes de almacenar el triángulo, es saber sí es una cara oculta o no, ya que en caso de serlo este triángulo no será almacenado, la manera para determinar sí una cara es oculta o no es mediante la observación de la orientación de sus vértices, si la orientación de los vértices vista desde el punto de observación esta en el sentido de las manecillas del reloj decimos que la cara es oculta.

Un detalle importante que hay que mencionar es el que se realiza en el arreglo VERTEX, ya que para cada vértice se tendrá un campo llamado CONEXION, el cual nos dirá con que otros vértices forman segmentos; esto se hizo con la idea de evitar redundancias, ya que si por ejemplo tenemos un vértice i que forma un segmento con el vértice j, no deberemos de tener en este mismo arreglo el vértice j que forme un segmento con el vértice i.

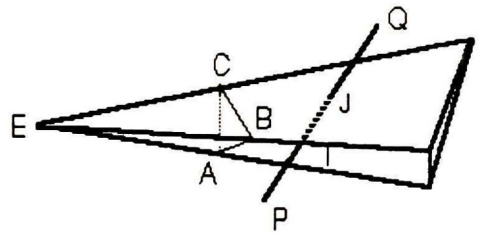
Una vez que se hallan realizado estos procesos previos, se procederá a dibujar al objeto el cual se realizará de la siguiente manera: Para cada vértice, se toman los vértices que están conectados a el para ir formando segmentos que se irán dibujando si estos son visibles, para determinar si un segmento es visible (o parte de el) se aplicará el algoritmo de ocultamiento de líneas que será explicado a continuación.

5.3.1.1.- ALGORITMO DE OCULTAMIENTO DE LINEAS

Como recordaremos anteriormente, cada cara del objeto será dividido en triángulos con vértices A,B,C. La tarea principal de este algoritmo es investigar sí el triángulo A,B,C oculta parcial o totalmente al segmento PQ.

Desde un punto de Observación E, se dice que el triángulo ABC oculta a R si el segmento ER intersecta al triángulo ABC en un punto interior, tanto al triángulo como al segmento y el triángulo ABC no oculta a R si R es un punto interior de ABC.

Este algoritmo lo que realiza primero es la construcción de un objeto llamado "Pirámide" que será formado por el punto de observación E cuyos planos pasan a través de los lados AB, BC y CA del triángulo (fig. 5.10). La Pirámide servirá para ilustrar las pruebas de visibilidad de un segmento con respecto a un triángulo; todos los puntos que estén enfrente del triángulo o fuera de la Pirámide son visibles con respecto al triángulo y cualquier punto en la Pirámide que se encuentre atrás del triángulo es invisible.



LA PIRAMIDE

Fig 5.10 La Pirámide.

La complejidad del algoritmo radica en el gran número de casos que se tienen que tratar. El algoritmo de ocultamiento de líneas realiza las siguientes pruebas:

Prueba 1

Si ambos puntos P y Q caen antes o en el plano ABC (no detrás de este), entonces PQ es visible (fig. 5.11), Esto sucede cuando:

$$n^*EP \leq h \text{ y } n^*EQ \leq h$$

donde n = [a b c].

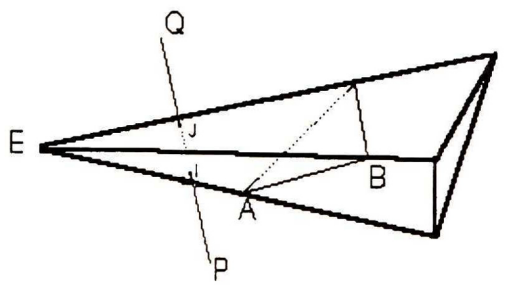


Fig 5.11

Prueba 2

Si la línea PQ cae fuera de la pirámide, PQ es visible (fig. 5.12).

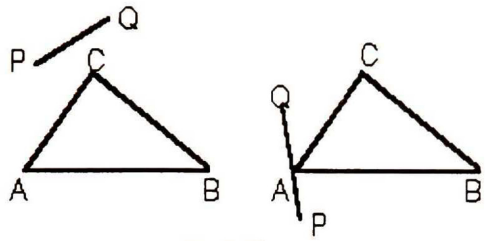


Fig 5.12

Prueba 3

Ahora calcularemos las intersecciones de la línea PQ con los planos EAB, EBC, EAC (fig. 5.13). Si PQ esta mas allá con respecto a un mismo lado del triángulo o si uno de ellos está mas allá de un lado y el otro está sobre el mismo lado, decimos que PQ esta fuera del triángulo y por lo tanto PQ es visible.

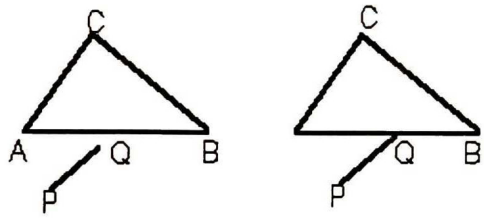


Fig 5.13

Prueba 4

Si P y Q están fuera de la pirámide (y las pruebas anteriores han fallado), y PQ cae atrás del triángulo entonces no es visible (fig. 5.14)

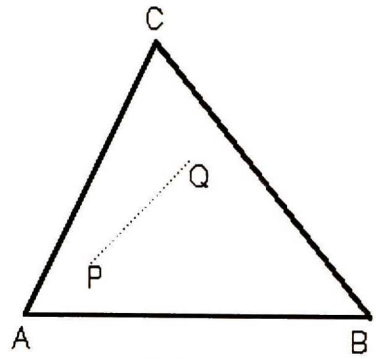


Fig 5.14

Prueba 5

Si un punto I, donde PQ intersecta a la pirámide, cae enfrente del triángulo, entonces PQ es visible (fig. 5.15)

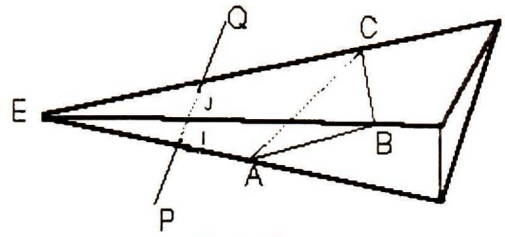


Fig 5.15

Prueba 6

Si las pruebas anteriores fallaron, significa que PQ intersecta a la pirámide atrás del triángulo. En esta prueba se calculan los puntos de intersección I,J (fig 5.4), IJ es invisible y ademas si:

Si P esta fuera de la pirámide o atrás del plano ABC, PI es visible.

Si Q esta fuera de la pirámide o atrás del plano ABC, JQ es visible.

Si P y Q caen fuera de la pirámide, entonces los puntos I,J coinciden (fig. 5.16).

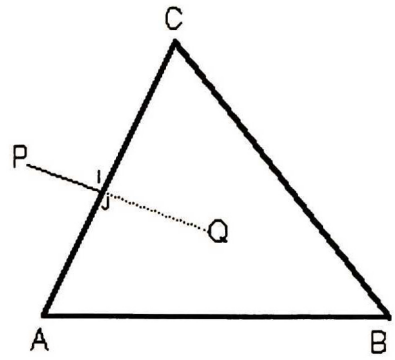


Fig 5.16

5.3.2.- ILUMINACION

El realismo de las imágenes en tercera dimensión depende de la simulación efectiva de los efectos de iluminación, la iluminación es usada para calcular la intensidad y los colores que posee la superficie. Con esto mejoraremos la calidad de la imagen pero no esperamos que se pueda desplegar el objeto exactamente como aparece en la realidad, lo que se desea es que la imagen de una apariencia de realismo. Por lo tanto la calidad de la imagen depende directamente de la efectividad del algoritmo de iluminación, el cual esta directamente relacionado al método para modelar el objeto.

La iluminación tiene dos principales elementos que son las propiedades de la superficie y las propiedades de la iluminación que caen en esta, la propiedad principal de una superficie es la refractancia la cual determinada la cantidad de luz que es reflejada. Si una superficie tiene diferentes refractancias de luz en diferentes ondas esta será coloreada. Si una superficie es texturizada o tiene un patrón de pintado en esta, la refractancia podra variar con la posición de la superficie. Otro propiedad de la superficie que juega un papel importante en la iluminación es la transparencia (una superficie que permite alguna luz que es transmitida atrás de esta). En la

iluminación de los objetos es muy importante las propiedades de la superficie para calcular la intensidad.

Debido a la limitante que se tiene en las PC's la iluminación que se realizó sobre los objetos fractales fue la más simple, ya que no se consideró la transparencia; el algoritmo usado para la iluminación lleva acabo los siguientes pasos:

1.- Definir el coeficiente de iluminación.
2.- Calcular el vector Normal de la superficie a iluminar.
3.- Obtener el coseno theta, el cual se encuentra entre el Vector Normal de la superficie y la luz de incidencia que cae sobre esta superficie (fig. 5.17).

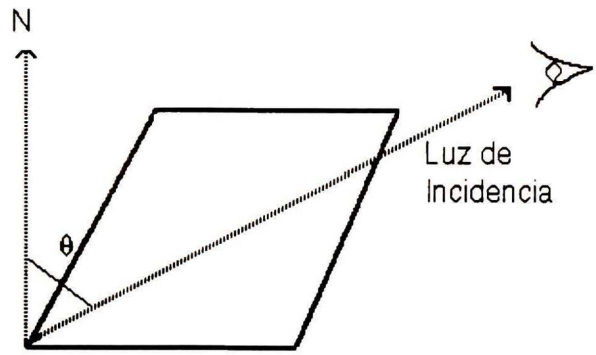


Fig 5.17

esta relación del coseno es conocida en la óptica como la ley del coseno y nos permite calcular la iluminación Sp para cualquier superficie p:

$$Sp = Cp * \cos(\text{Vector Normal de P / La luz de Incidencia})$$

Esta iluminación ofrece una muy pobre aproximación al efecto físico real pero tiene la ventaja de que es muy sencilla de calcular y la velocidad de iluminación es muy rápida.

CAPITULO 6

EDITOR DE PAISAJES

6.1 DESCRIPCION DEL PROBLEMA

Uno de los grandes problemas que se presentan en la mayoría de los editores gráficos, es la imposibilidad de generar figuras naturales tales como árboles, montañas, nubes, ríos etc.; ya que estos objetos no pueden ser especificados por la geometría Euclidiana y hay muchos problemas para poderlos dibujar.

Una alternativa para poder dibujar este tipo de figuras sería la de ir pintanto punto a punto la silueta de estos objetos; como se puede percibir ésta sería una mala solución, ya que la cantidad de datos que tendríamos que almacenar para cada uno de ellos sería bastante grande y el tiempo de graficación de cada una de estas figuras sería bastante lento, por lo tanto está alternativa estaría bastante restringida.

Otra alternativa para poder graficar estas figuras es la que ofrecen los fractales (la forma natural de dibujar objetos de la naturaleza), ya que con transformaciones afines (IFS), reglas de producción (gramáticas de Lindermayer) y movimiento Browniano se pueden representar árboles, montañas, ríos, nubes, etc.

Podemos decir que esta línea representa un buen camino para dibujar estos objetos naturales ya que su especificación es muy simple, sencilla y la cantidad de información almacenada es muy pequeña comparado con cualquier otro método que se puede utilizar para la generación de estos objetos.

6.2 EDITOR DE PAISAJES

6.2.1 DESCRIPCION GENERAL

El editor de paisajes fué diseñado para operar con la técnica famosa del pintor. Esta técnica básicamente nos dice que los objetos más lejanos se tendrán que pintar primero y luego los objetos mas cercanos y así sucesivamente hasta llegar a los objetos más cercanos.

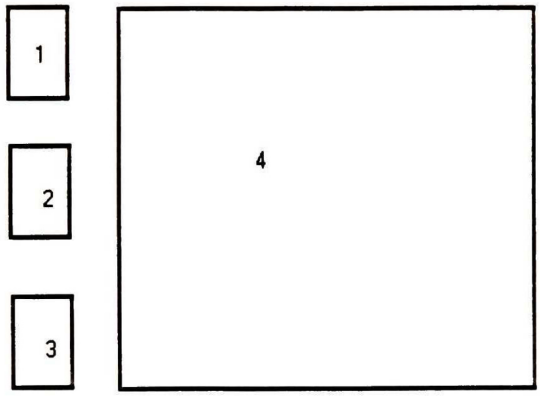
El editor de paisajes no ocupa toda la pantalla ya que el modo gráfico en el que se esta ejecutando, no permite más de una página a la vez, lo que representa un gran problema ya que si ocupáramos toda la pantalla, los menúes de diálogo, de color y de elección se tendrían que salvar en un archivo auxiliar, lo cual retardaría la velocidad del editor, debido a los accesos a disco que se tendrán que hacer para manejar los diversos menúes que presenta el editor de paisajes.

Otro problema que se presenta con este tipo de implementación de ventanas es el de los objetos (árboles, montañas, nubes, etc.), ya que estos puden salirse de la ventana de edición. Este problema se resolvió haciendo un recorte sobre la ventana de edición antes de pintar

cualquier objeto, lo cual nos asegura que el pintado de cualquier objeto no se saldrá de la ventana de edición.

El editor de paisajes fue hecho de modo interactivo con el usuario, para que este pueda editar con toda libertad el paisaje que desee.

El editor de paisajes consta de cuatro ventanas como se muestra en la fig. 6.1.



1.- Ventana del Menú Principal.
2.- Ventana de Diálogos.
3.- Ventana de Colores.
4.- Ventana de Edición.

Fig. 6.1 Panorama general del editor de paisajes.

1.- Ventana del Menú Principal

En esta ventana se manejan las opciones que tiene el editor de paisajes planos; además del menú principal existen varios submenúes que nos permiten elegir los objetos que deseamos generar.

Un esquema jerárquico de todos los menúes y submenúes es mostrado en la fig. 6.2.

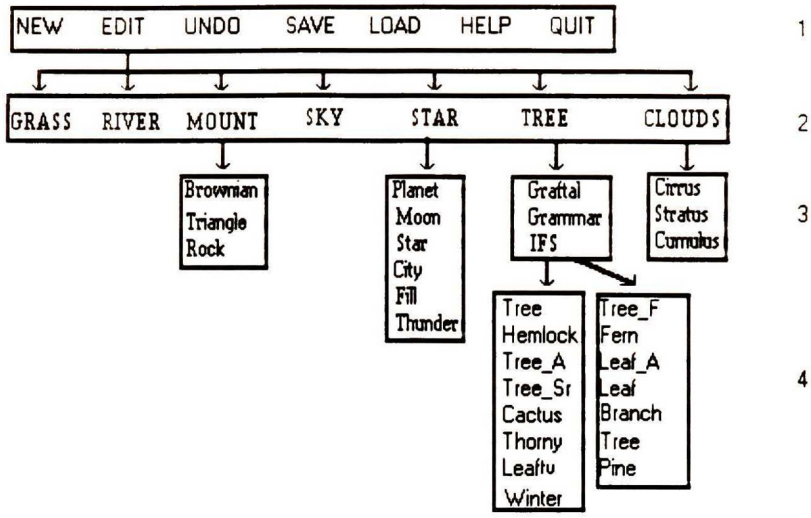


Fig. 6.2 Panorama jerárquico de las opciones del editor

2.- Ventana de Diálogos

La ventana de diálogos es el lugar donde se realiza la interacción del usuario con el editor, se proporcionan los parámetros (tamaño, ángulo, anchura, etc.) para la creación de los objetos fractales.

3.- Ventana de Colores

La ventana de colores nos permite seleccionar los colores que deseamos que tengan los objetos que generamos.

4.- Ventana de Edición

La ventana de edición, es el área destinada a la edición de los paisajes.

6.2.2 MENU PRINCIPAL

El editor de paisajes tiene el siguiente menú principal:

Al inicio siempre, el manejador del menú se posicionará en NEW. Una característica importante de este menú es la condicionalidad de sus opciones, es decir el manejador del menú no permitirá elegir una opción determinada (SAVE, EDIT, UNDO) si previamente no se ha elegido alguna opción diferente a éstas (a excepción de HELP).

Al inicio sólo podemos elegir una de las siguientes cuatro opciones:

ninguna otra opción podrá ser elegida, esto es con el fín de proteger al sistema de posibles errores del usuario, para poder tener acceso a las demás opciones del menú se tendrá que seleccionar una de las siguientes opciones:

A continuación daremos una descripción detallada del menú principal:

NEW

La opción NEW nos permite realizar las siguientes tareas:

a) Limpiar la ventana de edición para iniciar la edición de un nuevo paisaje.
b) Seleccionar el color de la paleta que se desee.

Nota:

Se recomienda al usuario que el color de la paleta no sea de color café, ya que este color se utiliza en la generación de árboles (Tronco).

EDIT

La opción EDIT es la parte más importante del sistema, ya que nos permite dibujar los diferentes fractales con que cuenta el sistema.

LOAD

La opción LOAD nos permite cargar un paisaje previamente salvado, para continuar con su elaboración. Si el nombre del archivo no se encuentra en el directorio corriente se mandará un mensaje de error.

SAVE

La opción SAVE nos permite salvar el paisaje que se generó y almacenarlo en un archivo. Si el nombre del archivo existe, este será reescrito con el paisaje corriente, por lo tanto se recomienda que el nombre del archivo que se proporcione no exista en el directorio corriente.

UNDO

La opción UNDO nos permite borrar el último objeto que se generó en el editor de paisajes, esta es una buena herramienta ya que si el objeto que se generó no corresponde con lo que se pensó este puede ser borrado.

Nota:

La opción UNDO queda inhabilitada cuando se desea borrar imágenes (paisajes) consecutivas.

HELP

La opción HELP nos proporciona información general para el manejo del sistema.

QUIT

La opción QUIT nos permite salir al sistema operativo, salvando la imagen del paisaje que se realizó en el archivo FILE_1.

6.2.3. DESCRIPCION DE LOS OBJETOS FRACTALES

A continuación se dara una descripción resumida de la manera de como se genera cada uno de los objetos fractales con los que cuenta el editor de paisajes.

GRASS(PASTO)

Técnica:

Gramáticas de Lindermayer.

Descripción:

Esta gramática es muy sencilla ya que solo consiste en ir pintando líneas con diferentes ángulos. La zona de pintado del pasto quedará determinada por la ventana definida por el usuario y el color del pasto tambien será eligido por el usuario.

RIVER (RIO)

Técnica:

Movimiento Browniano.

Descripción:

Se proporciona un número entero con el cual se generará la secuencia Browniana, esta secuencia Browniana quedará almacenada en un arreglo de tipo real; el tamaño del río Browniano quedará determinado por :

a) La ventana definida.
b) El ancho del río.

La técnica para simular el río Browniano consiste de tres aspectos:

a) Se pinta la trayectoria que determinó el movimiento Browniano.
b) Se pinta la misma trayectoria Browniana pero sumando el ancho que se le proporcionó.
c) Se rellenará la parte delimitada por las dos trayectorias Brownianas con el color elegido.

MOUNT (MONTAÑA)

BROWNIAN (MONTAÑA BROWNIANA)

Técnica:

Movimiento Browniano.

Descripción:

La montaña Browniana fué realizada de la siguiente manera.

Se proporciona un número entero con el cual se generará la secuencia Browniana, esta secuencia quedará almacenada en un arreglo de tipo real, el tamaño de la montaña browniana quedará determinada por:

a) La ventana definida.
b) El ancho de la montaña.

La técnica para simular la montaña Browniana es:

a) Se pinta la trayectoria que determinó el movimiento Browniano.
b) Se pinta la misma trayectoria Browniana pero sumándole el ancho que se proporcionó.
c) Se rellenará la parte delimitada por estas dos trayectorias Brownianas con el color que se proporcionó.

Esta técnica es muy rápida y sencilla, nada más que esta no presenta grandes detalles de las montañas, por lo tanto esta técnica es muy recomendable para las montañas que se encuentran alejadas del paisaje.

TRIANGLE (MONTAÑA POR TRIANGULACION)

Técnica:

Triangulación.

Descripción:

Esta montaña se realiza con la técnica de triangulación descrita anteriormente, este tipo de montañas presentan más detalles que las montañas Brownianas, por lo tanto este tipo de montañas se recomienda para montañas que no estén muy alejadas del paisaje. El truco para simular esta montaña es la mezcla de colores, ya que da una apariencia de fragmentación; el número de colores que se pueden combinar son tres y el tamaño de la montaña quedará determinado por los tres vértices que se proporcionen.

ROCK (PIEDRA POR TRIANGULACION)

Técnica:

Triangulación.

Descripción:

La piedra se realiza de la misma manera que la montaña por triangulación, con la excepción de que se proporcionan cuatro vértices para formar la piedra y no tres vértices como en el caso de la montaña por triangulación.

SKY (CIELO)

Técnica:

Auxiliar, no Fractal (Euclidiana).

Descripción:

Esta técnica se realizó con la idea de simular las estrellas que estan muy distantes. El color de las estrellas (puntos) es siempre de color blanco, el pintado de las estrellas se realiza de manera aleatoria en toda la pantalla, por lo cual se realiza un recorte de los puntos que queden fuera de la ventana de edición.

STAR

PLANET (PLANETA)

Técnica:

Auxiliar, no Fractal(Euclidiana).

Descripción:

El planeta fue hecho de la siguiente manera:

1.- Se pinta una circunferencia de acuerdo al radio que se proporcionó.
2.- Se rellena el interior de la circunferencia con el color que se eligió.
3.- El centro de la circunferencia quedará determinado por el punto inicial que se proporcionó.

Un detalle importante es que el color que se elige para el rellenado de la circunferencia deberá (recomendable) ser diferente al color del fondo; ya que si no, se rellenará toda la ventana del editor de paisajes.

MOON (LUNA)

Técnica:

Auxiliar, no Fractal(Euclidiana).

Descripción:

Su descripción es la misma que se dió en PLANETA, con el único cambio de que el rellenado siempre se hará de color blanco.

STAR (ESTRELLA)

Técnica:

Gramáticas de Lindermayer.

Descripción:

Esta estrella, es en realidad una curva fractal llamada curva de von Koch de nivel uno, el rellenado de la estrella siempre será de color blanco y el centro de la estrella es determinadó por la posición(START) que se proporcione.

CITY (CIUDAD)

Técnica:

Auxiliar, no Fractal(Euclidiana).

Descripción:

La ciudad se realizó de la siguiente manera:

1.- Se pintan puntos horizontales con un incremento entre cada punto de trece unidades, desde el inicio de la ventana (x's) hasta llegar al límite de la ventana.
2.- Se pintan puntos verticales con un incremento entre cada punto de trece unidades, desde el inicio de la ventana (y's) hasta llegar al límite de la ventana.
3.- Se pintan aleatoriamente mil puntos en la ventana definida.

el efecto de la ciudad se hizo con la idea de simular una ciudad vista desde una distancia lejana y a una cierta altura.

FILL (RELLENADO)

Técnica:

Auxiliar, no Fractal(Euclidiana).

Descripción:

El rellenado es una técnica que se realiza de la siguiente manera:

1.- Se eligen los límites del rellenado.
2.- El rellenado sera del color que se eliga.

La técnica del rellenado se hizo con la idea de simular llanuras, mares, etc.; el rellenado será en toda la ventana que se determinó.

THUNDER (RAYO)

Técnica:

Movimiento Browniano.

Descripción:

El rayo fué hecho por medio del movimiento Browniano así como las ramificaciones que aparecen en el rayo.

Se proporciona un número con el cual se genera la secuencia Browniana, esta secuencia quedará almacenada en un arreglo de tipo real; el tamaño del rayo Browniano

quedará determinada por el ventana definida. La simulación del rayo se realizan de la siguiente manera:

1.- Se pinta la trayectoria que determinó la secuencia Browniana.
2.- En cada paso del pintado de la trayectoria browniana, se genera un número aleatorio, si este número aleatorio es menor que 5,000 se pintará una ramificación del rayo de tamaño tambien aleatorio.

TREE (ARBOL)

GRAFTALES

Técnica:

Sistema-1L de Lindermayer.

Descripción:

Los Graftales son una técnica muy parecida a las gramáticas de Lindermayer, con la particularidad de que las reglas de producción se especifican de modo interactivo y además tienen una cota de crecimiento. Las producciones hechas por los Graftales quedarán almacenadas en un arreglo de tipo caracter. El número de generación de Graftales se restingió por el hecho de que la cadena que se va generando crece muy rápidamente; si esto llega a ocurrir se cortará la generación de Graftales (15,000 elementos).

GRAMMAR (GRAMATICAS DE LINDERMAYER)

Técnica:

Gramáticas de Lindermayer.

Descripción:

Los árboles que se pueden generar usando GL son de 8 tipos distintos:

1.- Tree (Arbol).
2.- Hemlock (Abeto).
3.- Tree_A (Arbol aereo).
4.- Tree_Sr (Arbol sin ramas).
5.- Cactus (Cactus).
6.- Thorny (Espinoso).
7.- Leafy (Frondoso).
8.- Winter (Invernal).

La técnica es la misma que se explicó en capítulos anteriores. Algunas características importantes de estos árboles son:

1.- El color del tronco(Café) de casi todos los árboles está ya definido, ya que el color del tronco de todos los árboles es café.
2.- El color de las ramas es el único que puede ser elegido.
3.- La posición del árbol quedará determinado por la ventana que se definió.

IFS (Sistemas de Funciones Iteradas)

Técnica:

Sistemas de Funciones Iteradas.

Descripción:

Los diferentes tipos de árboles que se pueden generar usando IFS son 7:

1.- Tree_F (Arbol Fractal).
2.- Fern (Helecho).

3.- Leaf_A (Hoja de otoño).
4.- Leaf (Hoja).
5.- Branch (Rama).
6.- Tree (Arbol).
7.- Pine (Pino).

La técnica usada es la misma presentada en capítulos anteriores, teniendo como particularidades los siguientes puntos:
1. El color del tronco es de color café como en el caso de las GL.
2.- El color que se puede variar es el color de las ramas del árbol.
3.- El tamaño del árbol quedará determinado por el tamaño de la ventana que se proporcionó.
4.- El número de iteraciones es de 9,000.

CLOUDS (NUBES)

Técnica:

Sistemas de Funciones Iteradas.

Descripción:

Esta técnica es la misma que se describió en capítulos anteriores y fué utilizada para los tres tipos de nubes:
a)Cirrus.
b)Estratus.
c)Cumulus.

El número de iteraciones para generar las nubes es de 9,000 y el tamaño de la nube esta determinado por la ventana definida.

6.2.4 RUTINAS ESPECIALIZADAS

6.2.4.1.- MANEJADOR

El editor de paisajes cuenta con un manejador de menúes y submenúes, el cual responde a las siguientes teclas:

↑.- Se mueve a la siguiente opción del menú que se encuentra arriba.
↓.- Se mueve a la siguiente opción del menú que se encuentra abajo.
<Enter> .- Ejecuta la opción del menú que se eligió.
<Esc>.- Regresa al menú o submenú previo.

6.2.4.2.- BORRADO

El sistema cuenta con una opción UNDO la cual nos permite borrar el último objeto que se dibujó en el editor de paisajes; esto se hizo con el fín de que si el objeto que se dibujó (último) no corresponde al objeto que se desea, este puede ser borrado y de esta manera no volver a repetir todo el dibujó que antes ya se había creado.

Para poder realizar este UNDO, el sistema cuenta con dos archivos auxiliares llamados FILE_1, FILE_2, que es donde se almacenan las imágenes del editor. El salvado de la pantalla se realiza automáticamente y lo hace de la siguiente manera:
1.- Antes de salvar el paisaje con el último objeto que se dibujó, se procederá a hacer una copia del archivo FILE_1 al archivo FILE_2; esto se hará por medio de una función de la biblioteca de Turbo C que nos permite ejecutar un comando de MS-DOS.

2.- Se limpia todo el exterior de la ventana del editor; con el fín de salvar solo la información que se encuentra dentro de la ventana del editor de paisajes.

3.- Se salvará toda la pantalla con sus respectivos atributos, es decir la información de cada pixel se irá almacenando en el archivo.

6.2.4.3.- GRAFICACION

El sistema cuenta con una serie de rutinas básicas para la graficación de los objetos fractales y no fractales, estas rutinas proporcionan una herramienta eficaz para la graficación y son:

OPENGRAPH
Parámetros de Entrada: Ninguno.
Parámetros de Salida: Ninguno.
Descripción:
Rutina que inicializa el modo gráfico.

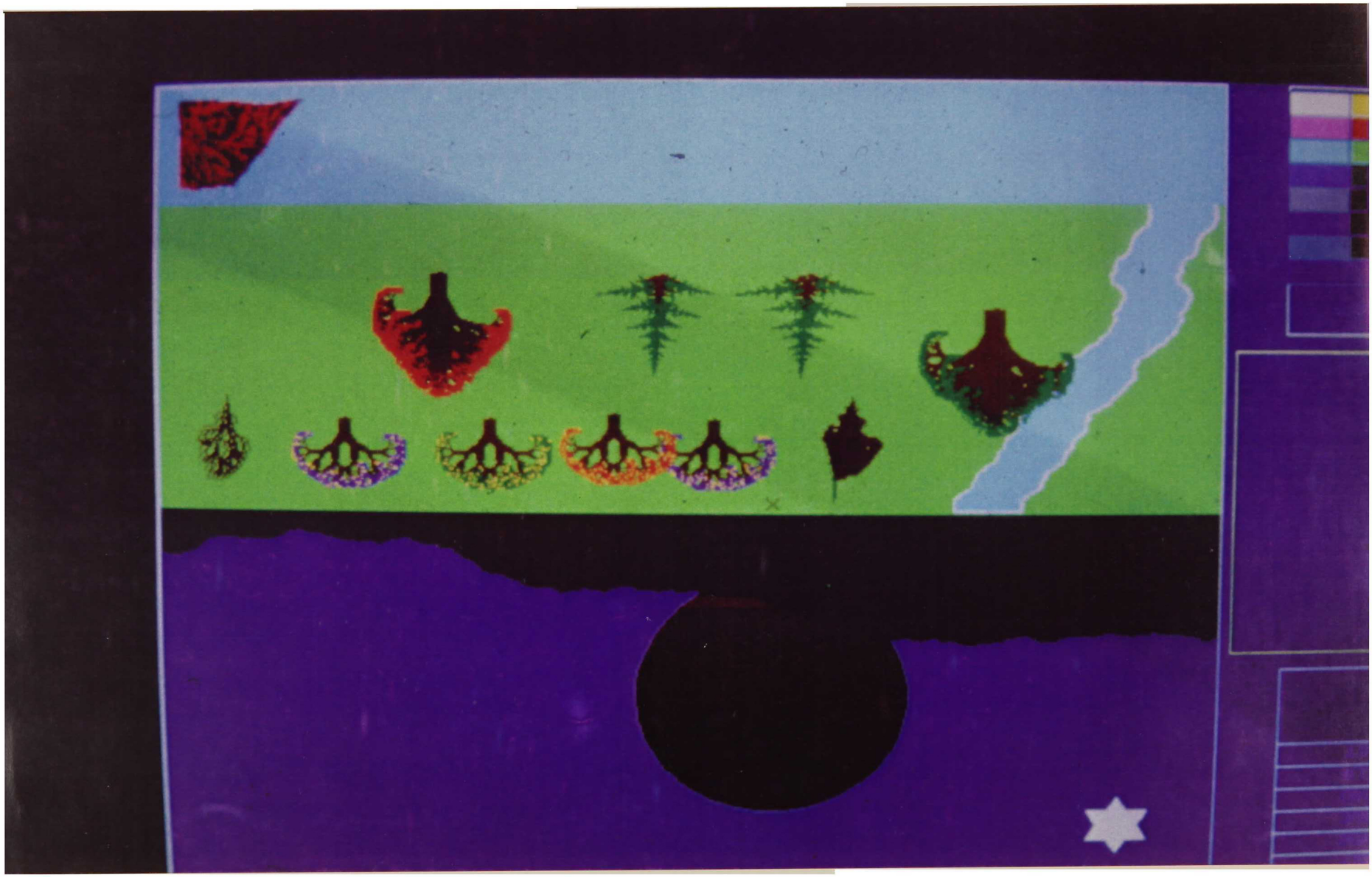
DRAWG
Parámetros de Entrada: X, Y
Parámetros de Salida: Ninguno.
Descripción:
Rutina que pinta una línea desde la posición del cursor a las coordenadas (X, Y) que se le proporcionó.

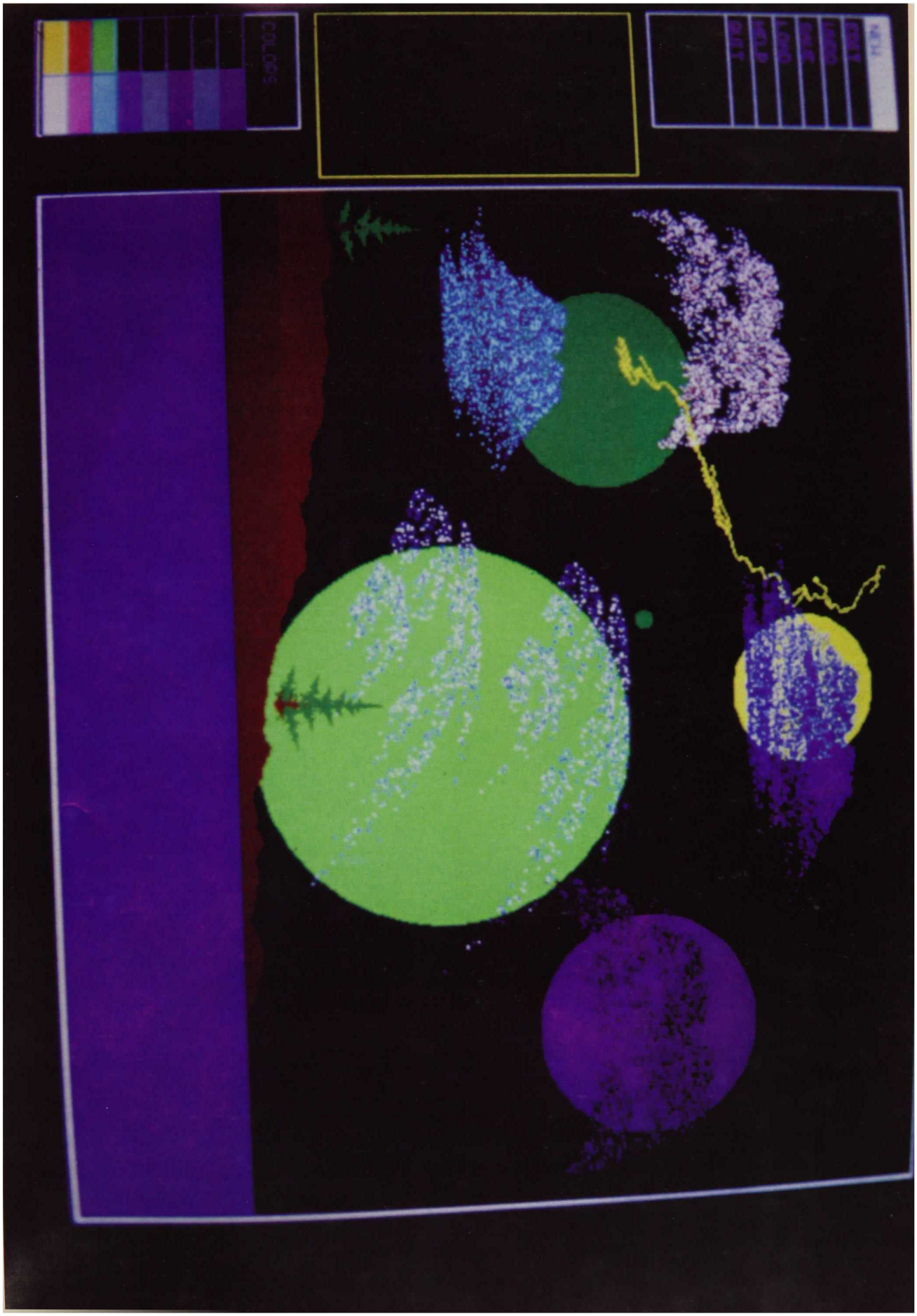
MOVEG
Parámetros de Entrada: X, Y
Parámetros de Salida: Ninguno.
Descripción:
Mueve el cursor a la posición (X, Y) que se le indica

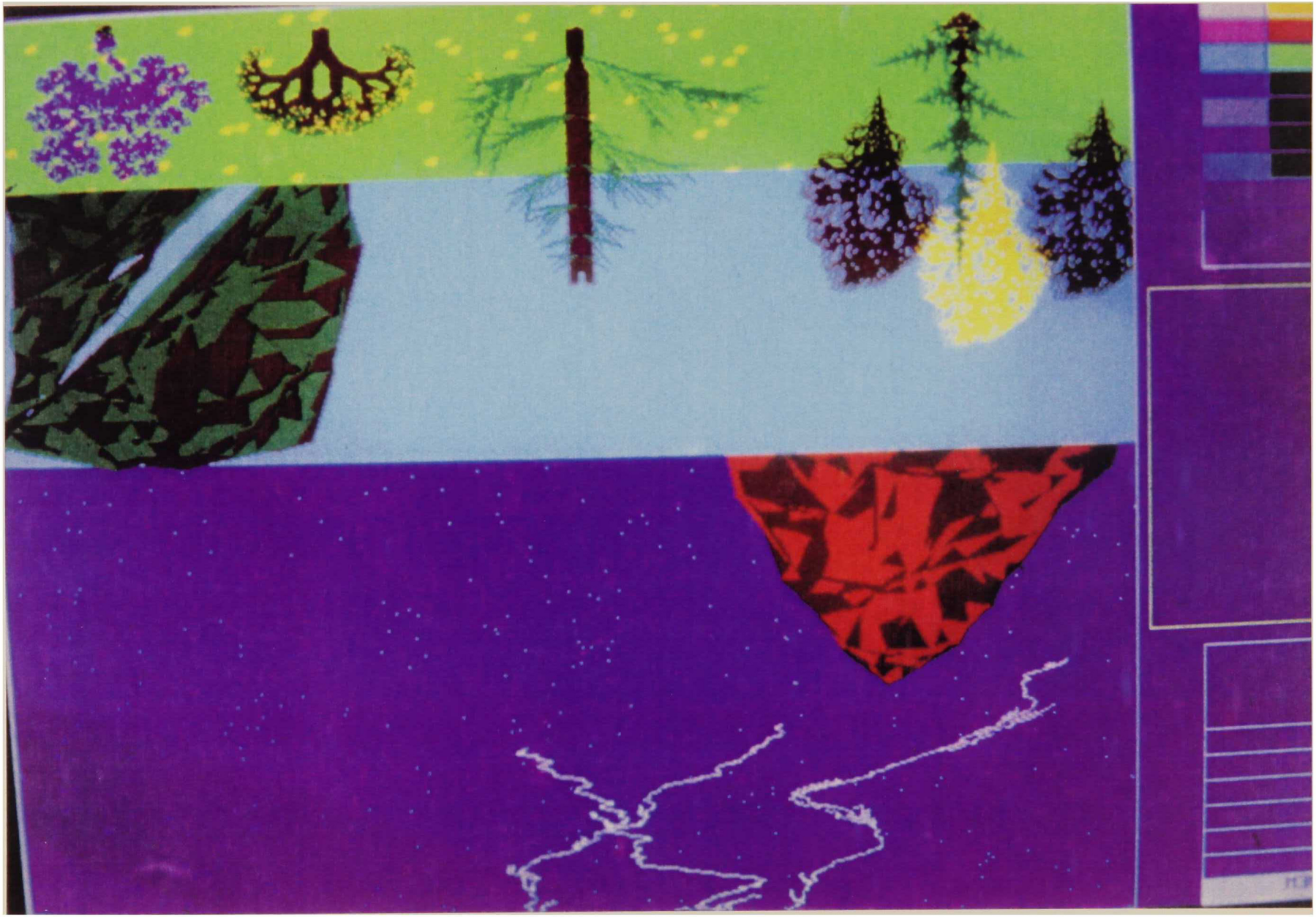
SCALE
Parámetros de Entrada: Xmin, Xmax, Ymin, Ymax.
Parámetros de Salida: Ninguno.
Descripción:
Pone los valores máximos y mínimos de X, Y de acuerdo a los valores que se le proporcione; estos valores son usados en las rutinas Drawg y Moveg.

LOCATE
Parámetros de Entrada: x1, y1, x2, y2.
Parámetros de Salida: Ninguno.
Descripción:
Define la ventana en donde se graficará.

FRAME
Parámetros de Entrada: Ninguno.
Parámetros de Salida: Ninguno.
Descripción:
Pinta la ventana en donde se esta graficando.




COLORS
RED
BLUE
GREEN
CYAN
MAG
YLW



6.3.- EJEMPLOS

A continuación se presentan los siguientes 2 ejemplos de paisajes que fueron realizados con el editor de paisajes.

CONCLUSIONES Y PERSPECTIVAS

CONCLUSIONES

Las conclusiones que he obtenido al finalizar esta tesis son las siguientes:

1.- Cuando el tipo de fractales que se desea obtener, no se tiene una idea clara de como contruirlo, la técnica que recomendaria usar es IFS, ya que esta técnica nos proporciona una interfaz para el diseño de fractales (método del Collage).

2.- Cuando se requiere generar fractales del tipo árboles, flores, etc., la técnica que mejor se adapta a este tipo es la gramática de Lindermayer.

3.- El movimiento Browniano es la técnica ideal para la generación de montañas y rocas, ya que su fundamentación teórica se acopla idealmente a la apariencia de una montaña o roca , lo mismo sucede con la técnica de Triangulación.

4.- Para poder generar fractales de muy alta calidad (iluminación, texturación, etc.) se requiere estaciones de trabajo minimamente, tales como SUN, HP, DEC, etc.

5.- Los Fractales son la manera más eficiente y cómoda de dibujar los objetos de la naturaleza tales como árboles, nubes, ríos, montañas, etc.

6.- El editor de paisajes, sienta las bases para la creación de escenarios de alta calidad, ya sea para películas o para comerciales.

7.- La explotación de los fractales, en algunos áreas como Exploración Petrolera, Música, medicina, etc., se encuentra en sus inicios, pero con perspectivas bastante alentadoras, en donde quiza dentro de algunos años podramos observar sus resultados.

PERSPECTIVAS

Sería bastante apropiado implementar al editor de paisajes las siguientes innovaciones:

1.- Poder realizar animación en cada uno de los objetos que se tiene en el editor.

2.- Aumentar el número de objetos que puedan ser seleccionados en el editor de paisajes.

3.- Añadir a los objetos iluminación y texturización para darle una apariencia más real.

4.- Aumentarle una opción de lectura de objetos ajenos al editor tales como: casas, automóviles, aviones, personas, animales, etc.

5.- La creación de los Fractales tridimensionales sienta las bases para la creación de un editor de paisajes tridimensional.

Por lo que respecta al editor de Fractales podrian implementarse las siguientes innovaciones:

1.- Tener las posibilidad de cambiar las transformaciones de los objetos e ir dibujando al objeto que se va generando.

2.- Que las transformaciones no solo sean cuadrados fijos si no que se puedan modificar.

APENDICE

A.1 MANUAL DE USUARIO DEL EDITOR DE FRACTALES

Los programas requeridos para la ejecución del editor de fractales son:

1.- EDI_FRA.EXE
2.- LITT.CHR
3.- EGAVGA.BGI

El equipo mínimo que se requiere para poder ejecutar Menú es:

1.- Una microcomputadora 8086 con monitor monocromático.
2.- 512k de memoria principal.
3.- Un disco flexible de 370K.

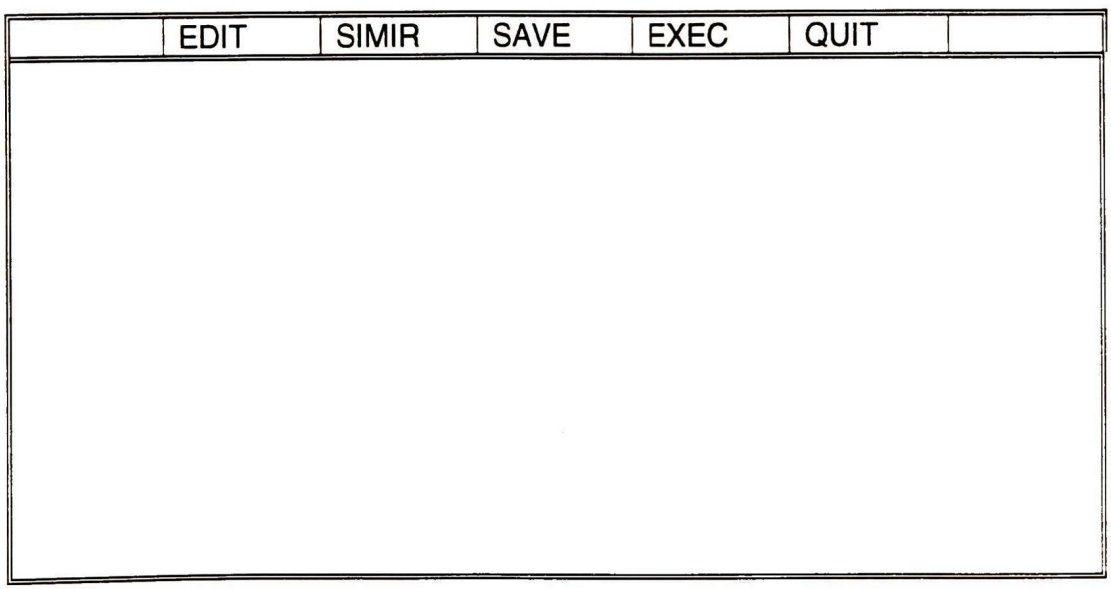
A continuación daremos una descripción breve de lo que puede considerarse un manual de usuario a través del siguiente ejemplo.

Supongamos que deseados obtener los códigos de las transformaciones afines del triángulo de Sierpinski:

Ordene al sistema operativo que ejecute EDI_FRA como se muestra a continuación:

C> EDI_FRA <Enter>

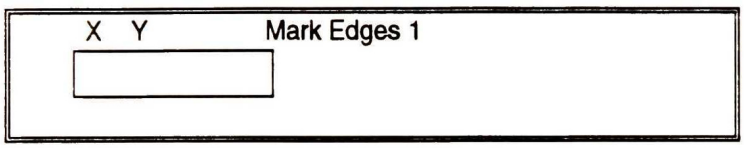
Aparecerá a continuación el menú principal



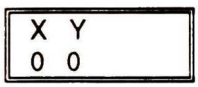
con las flechas (→ , ←) seleccione la opción EDIT y presione <Enter>, en la ventana de diálogo responda con el número 3 a la siguiente pregunta:

a) # Edges(3-20): 3.

después de esta pregunta aparecerá el siguiente mensaje en la ventana de diálogo

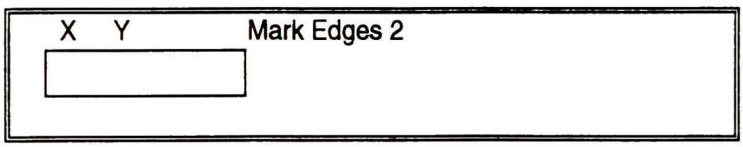


con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas (→ , ← , ↑ , ↓) posicione el cursor (X) en las siguientes coordenadas:

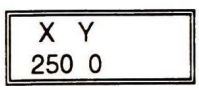


y oprima <Enter>.

Inmediatamente después de presionar <Enter> se desplegará el siguiente letrero en la ventana de diálogo:

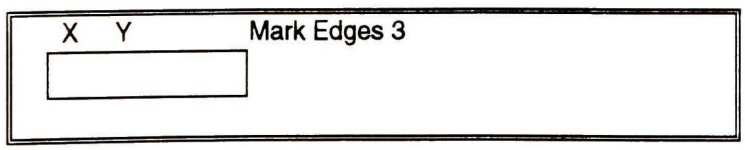


con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas (→ , ← , ↑ , ↓) posicione el cursor (X) en las siguientes coordenadas:

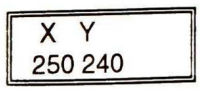


y presione <Enter>.

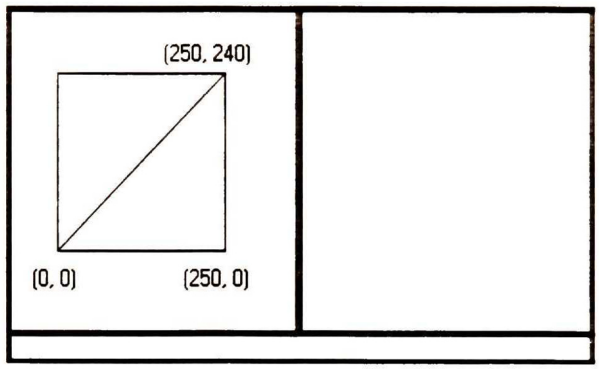
Después de presionar <Enter> se mostrará el siguiente letrero en la ventana de diálogo



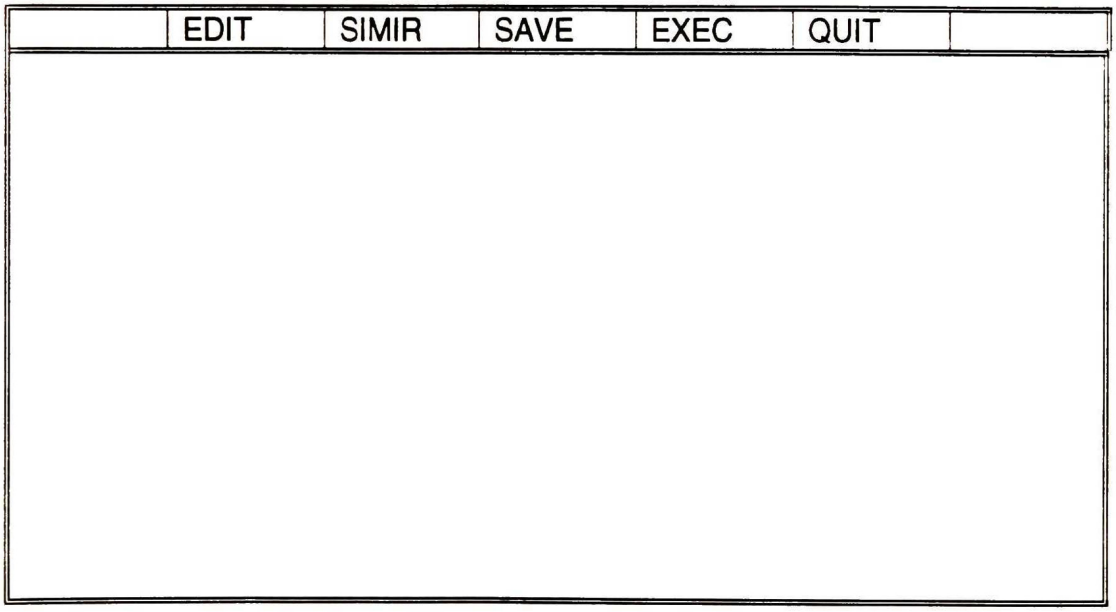
con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas (→ , ← , ↑ , ↓) posicione el cursor (X) en las siguientes coordenadas:



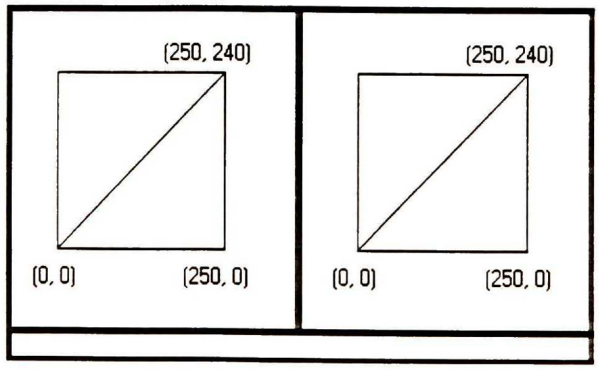
y presione <Enter>; Después de definir los vértices de la figura poligonizada, se visualizará en la ventana de la izquierda la siguiente figura:



y el control se regresará al menú principal



con las flechas ($\longrightarrow$, $\longleftarrow$) seleccione la opción SIMIR y presione <Enter>. Enseguida aparecerá la siguiente pantalla



en la ventana de diálogo se mostrará el siguiente letrero.

```
Mark 1er. Point
```

con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas ($\rightarrow$, $\leftarrow$, $\uparrow$, $\downarrow$) posicione el cursor (X) en las siguientes coordenadas:

```
X  Y
0  0
```

y presione <Enter>.

Enseguida se desplegará en la ventana de diálogo el siguiente letrero.

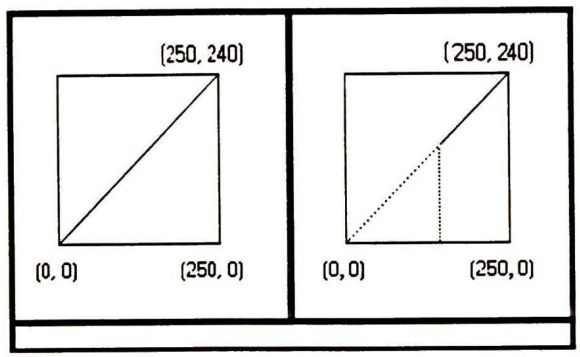
```
Mark 2do. Point
```

con las llaves (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas ($\rightarrow$, $\leftarrow$, $\uparrow$, $\downarrow$) posicione el cursor (X) en las siguientes coordenadas:

```
X  Y
125  0
```

y presione <Enter>.

Una figura autosimilar del triángulo de Sierpinski se dibujará en la ventana de la izquierda, donde el tamaño de esta figura quedará determinada por los puntos que se le proporcionó.



en la ventana de diálogo se presentarán las siguientes preguntas que deberá responder con la letra (n o N) y (y o Y) respectivamente:

Erase(Y/N) n
Continue(Y/N) y

continuando con este proceso en la ventana de diálogo se visualizará el siguiente letrero

```
Mark 1er. Point
```

con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas ($\rightarrow, \leftarrow, \uparrow, \downarrow$) posicione el cursor (X) en las siguientes coordenadas:

```
X  Y
125  0
```

y presione <Enter>. Enseguida en la ventana de diálogo se mostrará el siguiente letrero

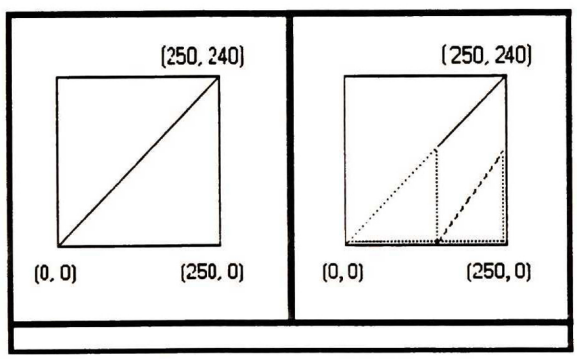
```
Mark 2do. Point
```

con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas ($\rightarrow, \leftarrow, \uparrow, \downarrow$) posicione el cursor (X) en las siguientes coordenadas:

```
X  Y
250  0
```

y presione <Enter>.

Una figura autosimilar del triángulo de Sierpinski se dibujará en la ventana de la derecha, donde el tamaño de esta figura quedará determinado por los puntos que se le proporcionó



en la ventana de diálogo aparecerán las siguientes preguntas que deberá responder con la letra (n o N) y (y o Y) respectivamente:

Erase(Y/N) n
Continue(Y/N) y

Continuando con este proceso en la ventana de diálogo se desplegará el siguiente letrero

```
Mark 1er. Point
```

con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas ($\rightarrow$, $\leftarrow$, $\uparrow$, $\downarrow$) posicione el cursor (X) en las siguientes coordenadas:

```
X  Y
125 120
```

y presione <Enter>.

Después de presionar <Enter> en la ventana de diálogo se mostrará el siguiente letrero

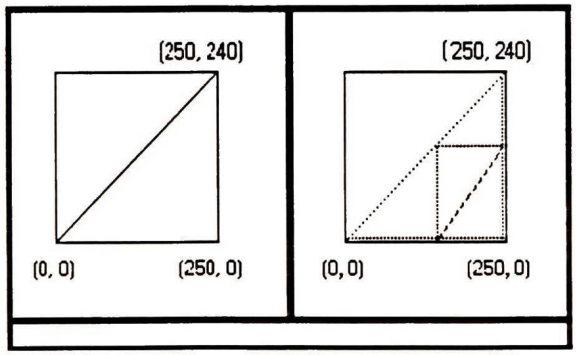
```
Mark 2do. Point
```

con las teclas (<Home>, <End>, <Pg Up>, <Pg Dn>) y las flechas ($\rightarrow$, $\leftarrow$, $\uparrow$, $\downarrow$) posicione el cursor (X) en las siguientes coordenadas:

```
X  Y
250 120
```

y presione <Enter>.

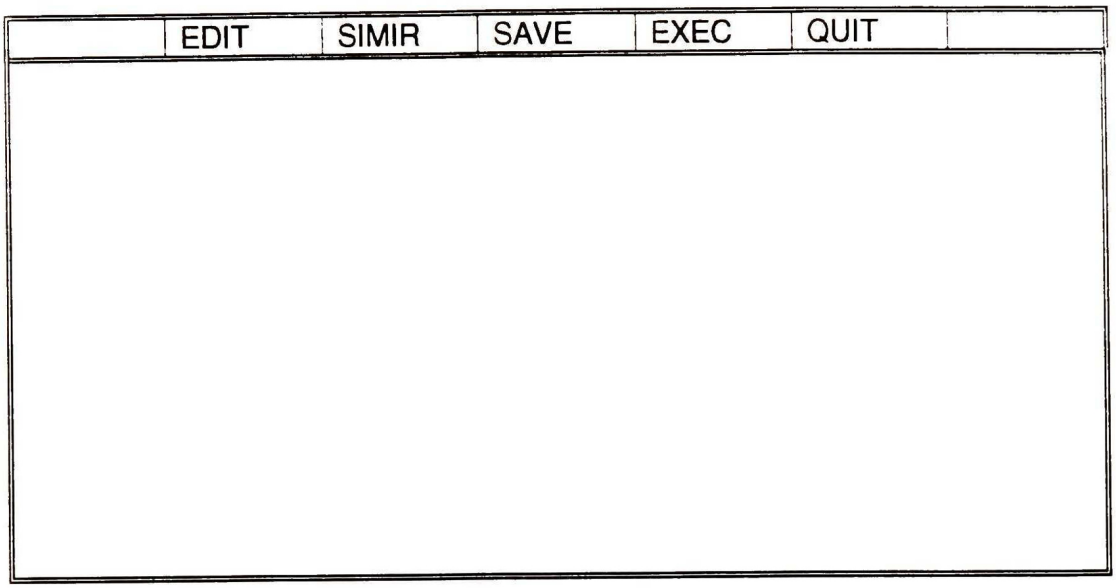
Una figura autosimilar del triángulo de Sierpinski se dibujará en la ventana de la derecha, donde el tamaño de esta figura quedará determinada por los puntos que se le proporcionó.



en la ventana de diálogo aparecerán las siguientes preguntas que deberá responder de la siguiente forma:

```
Erase(Y/N) n
Continue(Y/N) n
Edge I   1
Edge II  2
Edge III 3
```

después de responder a las preguntas anteriores, el control regresará al menú·principal

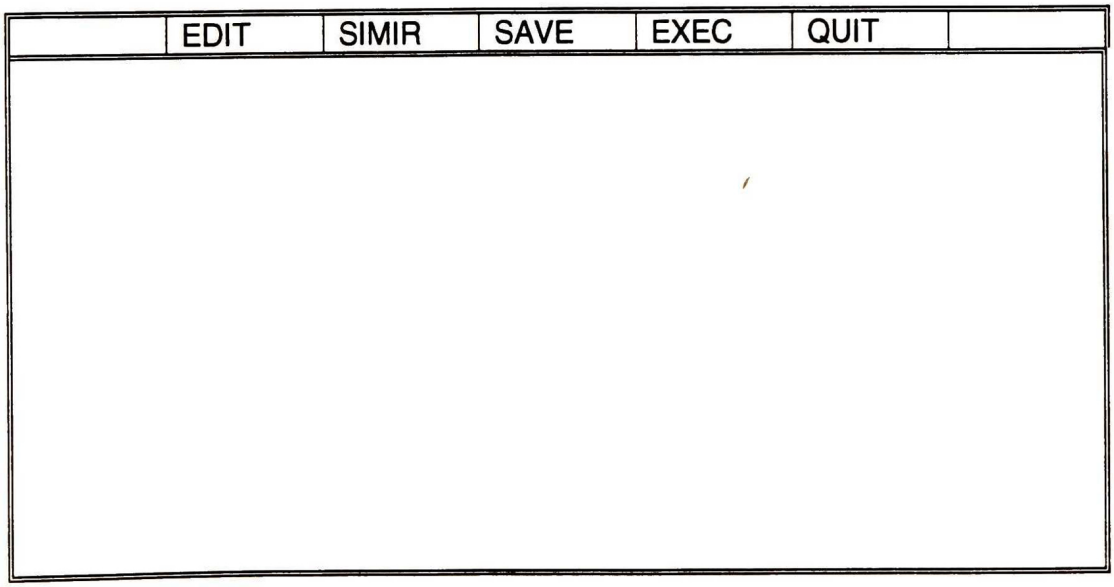


con las flechas (→ , ←) seleccione la opción SAVE y presione <Enter>.

En la ventana de diálogo proporcione el nombre del archivo en donde se almacenaran los códigos de las transformaciones afines del triángulo de Sierpinski.

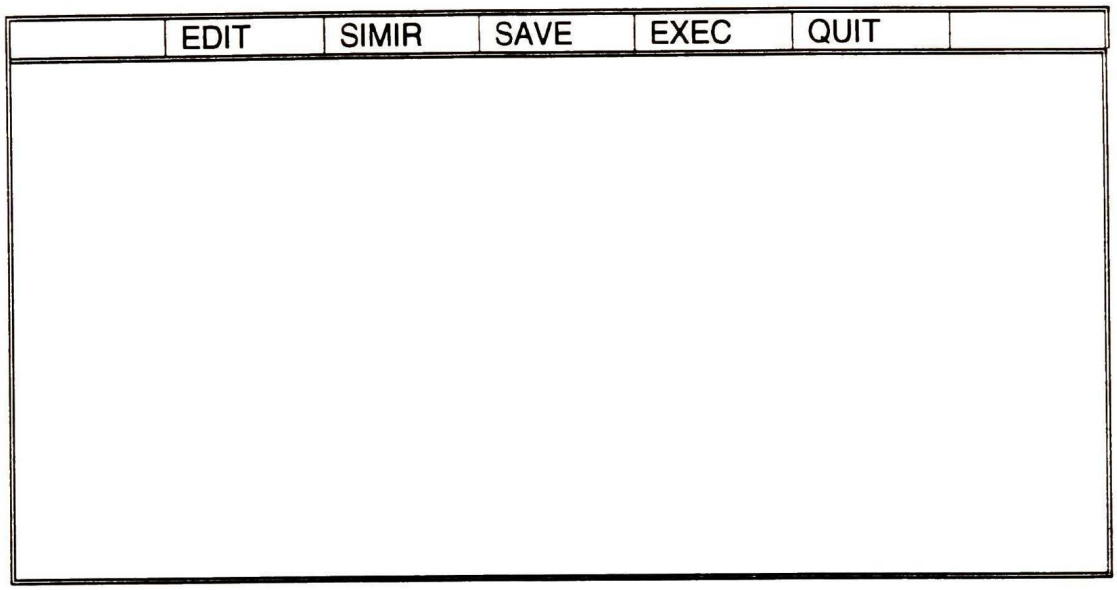
File Name: Sierpin <Enter>.

y enseguida se retornará el control al menú principal.

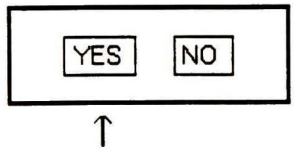


con las flechas (→ , ←) seleccione la opción EXEC y presione <Enter>.

La figura resultante del triángulo de Sierpinski será presentada en la Fig. A.2.1. El control será regresado al menú principal



con las flechas (→ , ←) seleccione la opción QUIT y presione <Enter> para salir al sistema operativo. En la ventana de diálogo aparecerá la siguiente pregunta:



con las flechas (→ , ←) seleccione YES y saldrá al sistema operativo.

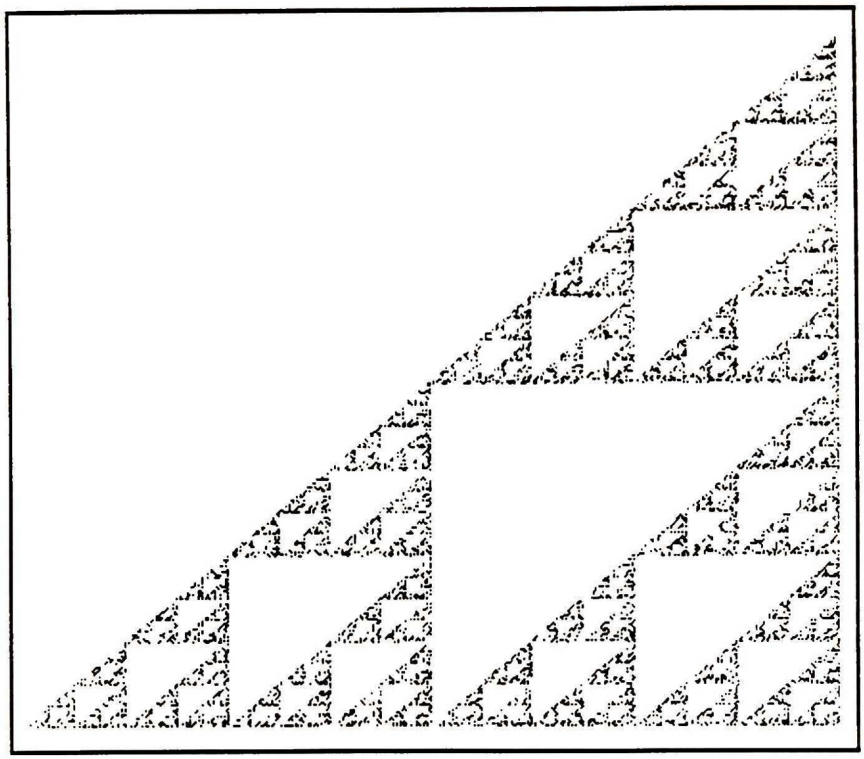


Fig. A.2.1.

A.2 MANUAL DE USUARIO DEL EDITOR DE PAISAJES

Los programas requeridos para la ejecución del editor de paisajes son los siguientes:

1.- EDI_LAND.EXE.
2.- LITT.CHR.
3.- EGAVGA.BGI.

El equipo mínimo que se requiere para poder ejecutar Edi_Land es:

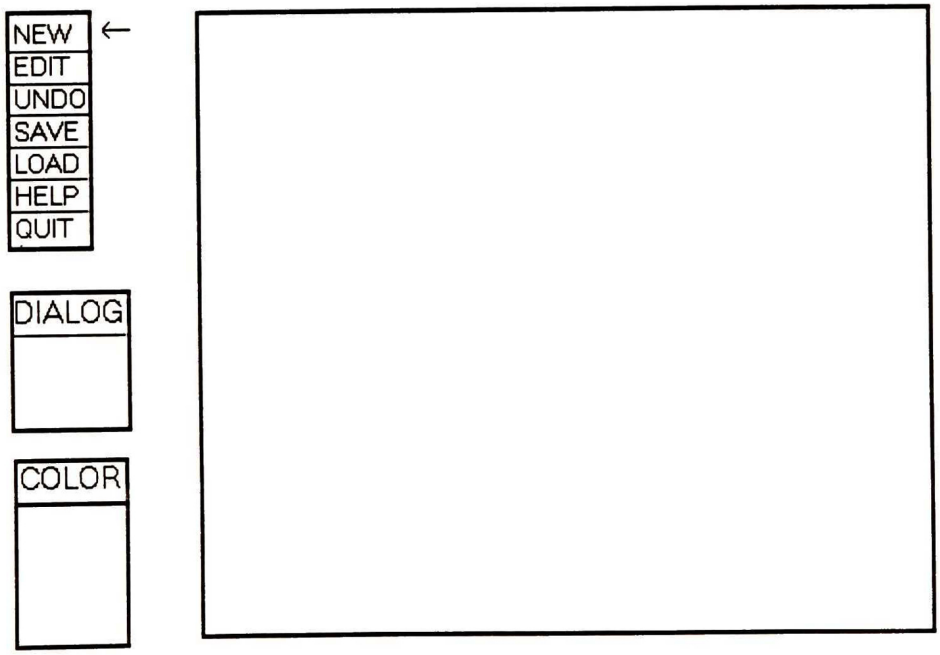
1.- Microcomputadora PC-AT(286) con monitor VGA a color
2.- 640k de memoria principal.
3.- Disco duro de 10 Megabytes.

A continuación daremos una descripción breve de lo que puede considerarse un manual de usuario através del siguiente ejemplo. Supongamos que deseamos graficar un paisaje con estrellas, montañas, nubes, árboles y un planeta.

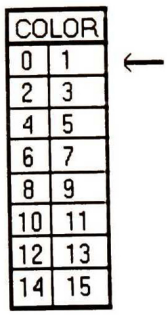
Ordene al sistema operativo ejecutar EDI_LAND como se muestra a continuación:

C> EDI_LAND <Enter>

A continuación el menú principal se desplegará



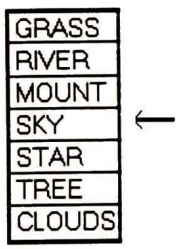
con las flechas ($\uparrow, \downarrow$) seleccione la opción NEW para inicializar la edición de un nuevo paisaje. En la ventana dialog aparecerá un letrero "Palette Color" lo cual indicará que deberá seleccionarse el color de la paleta que se desee.



0.- Negro.
1.- Azul Marino.
2.- Verde.
3 - Morado.
4.- Rojo.
5.- Violeta Oscuro
6.- Café
7.- Gris.
8.- Gris Opaco.
9.- Azul Rey.
10.- Verde Pistache.
11.- Azul Cielo.
12.- Anaranjado.
13.- Violeta Claro.
14.- Amarillo.
15.- Blanco.

seleccione con las flechas ($\rightarrow$, $\leftarrow$, $\uparrow$, $\downarrow$) el color azul marino y presione <Enter>. Después de realizar la elección del color de la paleta, seleccione EDIT <Enter>

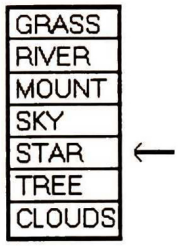
una vez hecho esto aparecerá el siguiente submenú.



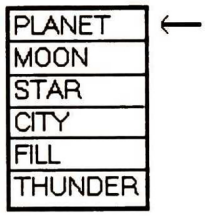
Seleccione SKY <Enter> y responda a la siguiente pregunta como se indica abajo:

#Ite (1-9999): 900 <Enter>.

Seleccione STAR <Enter>



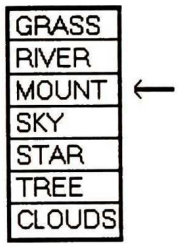
y de inmediato se verá el siguiente submenú



seleccione PLANET <Enter> y responda a las siguientes preguntas como se muestra a continuación:

START X= 250 Y= 390. <Enter>.
RATIO 80 <Enter>.
COLOR Violeta Oscuro <Enter>.

Seleccione MOUNT <Enter>



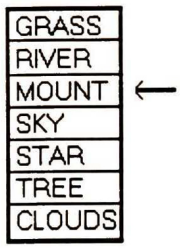
y enseguida el siguiente submenú·es desplegado

Seleccione TRIANGLE <Enter> y responda a las siguientes preguntas como se muestra a continuación:

Edge I X= 224 Y= 240 <Enter>
Edge II X= 0 Y= 240 <Enter>
Edge II X= 110 Y= 371 <Enter>
Color I Café <Enter>
Color II Rojo <Enter>

Color II Anaranjado <Enter>
Depth 3 <Enter>

Seleccione MOUNT <Enter>

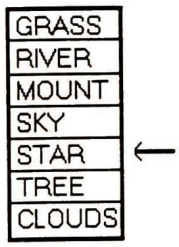


y enseguida se mostrará el siguiente submenú.

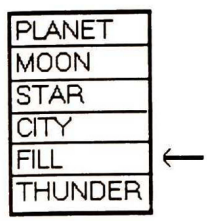
Seleccione TRIANGLE <Enter> y responda a las siguientes preguntas como se indica abajo:

Edge I X= 224 Y= 230 <Enter>
Edge II X= 360 Y= 410 <Enter>
Edge II X= 542 Y= 230 <Enter>
Color I Café <Enter>
Color II Rojo <Enter>
Color II Anaranjado <Enter>
Depth 3 <Enter>

Seleccione STAR <Enter>



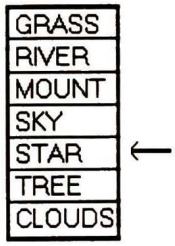
y se desplegará el siguiente submenú.



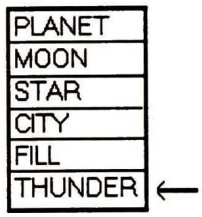
Seleccione FILL <Enter> y responda a las preguntas como se muestra a continuación:

TOP	X = 10 Y= 250. <Enter>
BOTTOM	X = 10 Y= 0. <Enter>
COLOR	Verde Pistache. <Enter>

Seleccione STAR <Enter>



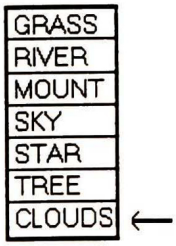
una vez hecho se tendrá en la pantalla el siguiente submenú.



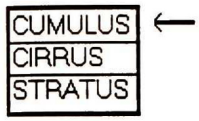
Seleccione THUNDER <Enter> y responda a las siguientes preguntas como se muestra a continuación:

SEED	100 <Enter>
WIDTH	4 <Enter>
COLOR	WHITE. <Enter>
TOP	X = 150Y= 476. <Enter>
BOTTOM	X = 290Y= 350. <Enter>

Seleccione CLOUDS <Enter>



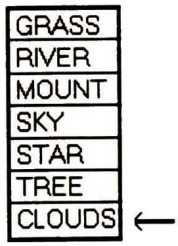
y se mostrará el siguiente submenú.



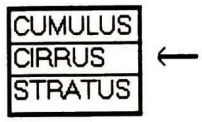
Seleccione CUMULUS <Enter> y responda a las siguientes preguntas como se muestra a continuación:

TOP X= 100 Y= 430 <Enter>
BOTTOM X= 190 Y= 345 <Enter>
COLOR I Azul Rey. <Enter>
COLOR II Azul Cielo. <Enter>

Seleccione CLOUDS <Enter>



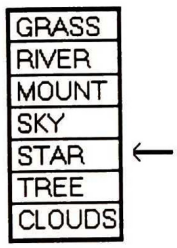
y se desplegará el siguiente submenú.



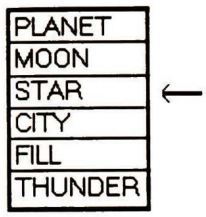
Seleccione CIRRUS <Enter> y responda a las siguientes preguntas como se muestra a continuación:

TOP X= 150 Y= 450 <Enter>
BOTTOM X= 360 Y= 360 <Enter>
COLOR I Azul Cielo. <Enter>
COLOR II Azul Rey. <Enter>

Seleccione STAR <Enter>



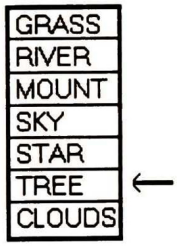
y se desplegará el siguiente submenú.



Seleccione STAR <Enter> y responda a las siguientes preguntas como se muestra a continuación:

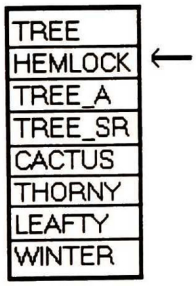
STAR X= 24 Y= 460 <Enter>
SIZE 6 <Enter>

Seleccione TREE <Enter>



del siguiente submenú seleccione GRAMMAR <Enter>

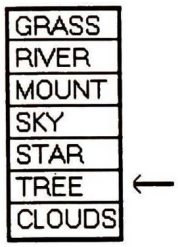
y enseguida se mostrará el siguiente submenú.



Seleccione HEMLOCK <Enter> y responda las siguientes preguntas como se muestra a continuación:

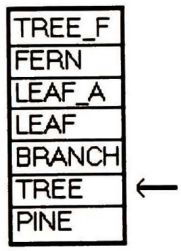
DEPTH 6 <Enter>
SIZE 5 <Enter>
TRUNCK 50 <Enter>
WIDTH 20 <Enter>
START X= 130 y= 80. <Enter>

Seleccione TREE <Enter>



del siguiente submenú·seleccione IFS <Enter>

y enseguida aparecerá el siguiente submenú



Seleccione TREE <Enter> y responda a las siguientes preguntas como se muestra a continuación:

TOP	X= 224 Y= 220 <Enter>
BOTTOM	X= 334 Y= 50 <Enter>
COLOR	VERDE <Enter>

oprima la tecla <Esc> para salir de este submenú y después seleccione SAVE <Enter>

el editor pedirá el nombre del archivo donde se almacenará la figura editada, responda como se muestra a continuación:

FILE NAME : PAI <Enter>

El paisaje resultante es mostrado en la Fig A.2.2. seleccione QUIT <Enter> para salir del editor de paisajes.

Fig A.2.2

BIBLIOGRAFIA

[1] Amburn, P., Grant, E., Whitted, T.
Managing geometric complexity with enhanced procedural methods
Computer Graphics, Agosto.
1986.

[2] Aono, M., and Kuni, T. L.
Botanical tree image generation.
IEEE Computer Graphics and Applications 4, Pag 10-34.
1984.

[3] Backus, J. W.
The syntax and semantics of the proposed international algebraic language of the
Zurich.
1959.

[4] Barnsley Michael.
Fractals Everywhere.
Academic Press.
1990.

[5] Barnsley M. F. , Elton J. ·
A new class of Markov processes for image encoding.
Journal of Applied Probability.
1986

[6] Barnsley, Michael F. , Ervin V., Hardin D. and Lancaster J.
Solution of an inverse problem for fractals and other sets.
Proceedings of the National Academy of Sciences, 83.
1985.

[7] Butz, A. R.
Converge with Hilberts space filling curve.
Journay Comp. System Science No. 3, Pag. 128-146)
1969.

[8] Butz, A. R.
Space filling curves and matematical
Information and Control 12, Pag. 315-320
1968.

[9] Chapa, Vergara S.
Integral de Wiener y su estimación Numérica.
Tesis Profesional ESFM (IPN)
México, 1973

[10] Chomsky N.
Three models for the description of language.
IRE Trans on Information Theory 2, Pag. 113-124.
1956.

[11] Crilly A. J., Earnshaw R. A. , H. Jones.
Fractals and Chaos.
Springer-Verlag.
1991.

[12] Dekking F. M.
Recurrent sets: A fractal formalism.
Report 82, Pag 32, Delft University of Technology.
1982.

[13] Demko Stephen.
Construction of Fractal Objects with IFS.
ACM Siggraph, Junio.
1988.

[14] Dewdney A. K.
Of fractal mountains, graftal plants and other computer graphics at pixar.
Scientific American, Diciembre, Pag. 14-18.
1986.

[15] Elton, J.
An ergodic theorem for iterated maps.
Journal of Ergodic Theory and Dynamical Systems.

[16] Feder, J.
Languages of enconded line patterns
Information and Control, Vol. 13, Pag. 230-244.
1968.

[17] Fournier, A., Fussell, D. and Carpenter L.
Computer rendering of stochastic models.
Comm. of the ACM, Pag 371-384.
1982.

[18] Frijters D. and A. Lindermayer.
A model for the growth and flowering of Aster novae angliae on the basis of table L-systems.
Lecture Notes in computer Science, Vol 15 Pag 24 -52.
Springer-Verlag
1974.

[19] Gardner, Martin.
Wheels, Life and Other Mathematical.
W. H. Freeman and Company.
1983.

[20] Goldberger Ary L., David R. Rigney and Bruce J. West.
Chaos and Fractals in Human Physiology.
Scientific American, Febrero, Pag. 36 -41.

1990.

[21] Hartmut Jürgens , Heinz-Otto Peitgen and Dietmar Saupe.
The Language of Fractals.
Scientific American, Agosto, Pag 40 - 47.
1990.

[22] Hogeweg P. and Hosper B.
A model study on biomorphological description.
Pattern Recognition, Pag. 165-179.
1974.

[23] Ivey Mark and Melcher Richard A.
Science & Technology.
BusinessWeek.
1991.

[24] Kawaguchi, Y.
A morphological study of the form nature.
Computer Graphics, Julio, Vol. 16.
1982.

[25] King Su Fu.
Syntatic Pattern Recognition and Aplications.
Digital Pattern recongnition.
1980.

[26] Langston P. S.
An experiment in music generation.
Bell Laboratories.
1986.

[27] Ledley R. S.
High-speed automatic analysis of biomedical pictures.
Science, Vol. 146, Pag. 216 -223.
1964.

[28] Mandelbrot B. B.
The Fractal Geometry of Nature.
W. H. Freeman.
1982.

[29] Mandelbrot B. B. and J.W. Van Ness
Fractional Brownian motion, fractional noises and applications
SIAM Review 10,4, Pag. 422-437.
1968.

[30] Meakin, P. A.
A new model for Biological pattern formation
J. Theor. Biol., Vol. 118, Pag. 101-113
1986.

[31] Miller, G. S. P.
The definition and rendering of terrain maps.
Computer Graphics, Pag. 39-48.
1986.

[32] Naokazu Yukoya , Kazuhiko Yamamoto y Noboru Funakubo
Fractal-Based Analysis and Interpolation of 3D Natural Surfaces and their Application to terrain modeling.
Computer Vision , Graphics, and Image Processing, Vol 46, Pag. 284 -302
1989.

[33] Narasimhan, R.
A linguistic approach to pattern recognition.
Report 21, Digital Computer Laboratory.
1962.

[34] Oppenheimer, P. E.
Real time design and animation of fractal plants and trees.
Computer Graphics, Agosto, Vol. 20.
1986.

[35] Peitgen-Saupe.
The Science of Fractal Images
Springer-Verlag.
1988.

[36] Przemyslaw Prusinkiewicz and James Hanan
Lecture Notes in Biomathematics, Vol. 79.
Springer-Verlag.
1989.

[37] Siromoney R. and Subramanian K. G.
Space-filling curves and finite graphs
Lecture Note in Computer Science, Vol. 153.
Springer- Verlag.
1983.

[38] Smith , A. R.
Plants, fractals and formal languages.
Computer Graphics Vol. 18, Pag 1-10.
1984.

[39] Szilard A. L. and Quinton R. E.
An interpretation for DOL systems by computer graphics.
The Science Terrapin 4, Pag 8-13.
1979.

[40] Voss, R. F.
Random fractal forgeries
Fundamental Algorithms for Computer Graphics.
Springer-Verlag.
1985.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL I.P.N.

EL JURADO DESIGNADO POR LA SECCION DE COMPUTACION DEL DEPARTAMENTO DE INGENIERIA ELECTRICA, APROBO EL DIA 07 **DEL MES DE** MAYO **DEL AÑO DE** 1993 **, EL TRABAJO DE TESIS.**

TECNICAS PARA EL MODELADO DE FRACTALES

DESARROLLADO POR EL ALUMNO:

LUIS ROBERTO GUTIERREZ ROMERO

DR. SERGIO V. CHAPA VERGARA
Jefe de la Sección de Computación

DR. JAIME RANGEL MONDRAGON
Profesor e Investigador del Centro de Ingeniería Computacional del Instituto Tecnológico, Campus Cuernavaca

DR. GUILLERMO MORALES LUNA
Profesor Titular de la Sección

Ave. Instituto Politécnico Nacional 2508 Col. San Pedro Zacatenco México, D.F. C.P. 07300
Tels: 752-0677 y 754-0200 Telex: 1772826 PPTME Fax: 752-0590 y 586-6290

AUTOR GUTIERREZ ROMERO, LUIS ROBERTO

TITULO TECNICAS PARA EL MODELADO DE FRACTALES

CLASIF. XM 93.12 RGTRO. BI-13509

NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
Dr. Sergio Chapa	10/03/95	14/3/95
Ju Shiguang	7/14/95	16/viii/95
Dr. Guillermo Morales	13-12-95	13/DIC/95
Ju Shiguang	9/5/96	23/5/96
Noe Sierra R.	17/7/96	7/OCT/
Ju Shiguang	3/1/97	31
Ju Shiguang	26/5/9	
Miguel A. Fajardo Ortiz	2	
Lopez Lopez Jose Luis		
Ana Mendoza C		
Juan A. Tiscareño		