APLICACIÓN DE LA LINGÜÍSTICA MATEMÁTICA A LA GENERACIÓN DE PAISAJES.

Fernando Galindo Soria
UPIICSA IPN
Calle de Te #950 , Col Granjas México, D.F. , 08400 México
Diciembre de 1992
fgalindo@ipn.mx www.fgalindosoria.com

Ir a la Página de la Ecuación de la Naturaleza S->e*S* www.fgalindosoria.com/ecuaciondelanaturaleza/

Ciudad. de México 1992
Construcción de la Página 29 de abril del 2001
Ultima modificación 30 de Julio del 2005

RESUMEN

Dentro del área de tratamiento de imágenes existe una gran cantidad de problemas en los que se requiere manejar objetos de la naturaleza como: arboles, nubes, estrellas y montañas, ya sea, para generar paisajes o construir animaciones de crecimiento de arboles o nubes entre otros.

Desde hace algún tiempo y en particular desde el surgimiento de la teoría de Fractales se han desarrollado métodos y técnicas orientados a resolver este tipo de problema y ya se cuenta con una gran cantidad de herramientas para resolverlos. Ahora bien, comúnmente se utilizan técnicas distintas para resolver los diferentes problemas, el método que permite generar las montañas no se aplica para generar árboles y por un lado se tienen métodos para generar nubes, por otro estrellas o arboles y por otro peces o caracoles, siendo cada método diferente y especifico y la construcción de un paisaje involucra el uso de múltiples métodos diferentes.

Por lo que, en este trabajo se presenta una herramienta general basada en la Lingüística Matemática y los fractales y en particular en una ampliación semántica de las Gramáticas Generativas, especificamente se presenta una ecuación lingüística de la forma S-->a*S*, con la cual se puede representar la estructura de múltiples elementos de la naturaleza, incluyendo arboles, nubes, estrellas, montañas, caracoles y ríos y que proponemos como una de las ecuaciones fundamentales de la naturaleza.

Dado que una regla lingüística o producción es fácil de representar mediante un programa de computo, se mostrara como representar la ecuación S-->a*S* mediante un pequeño programa, al que cambiándole el valor de un parámetro puede generar por ejemplo un bosque de helechos, o una nube en forma de perro, finalmente se muestra como con la misma rutina y simplemente cambiándole parámetros se pueden construir múltiples tipos de paisajes incluyendo: montañas nevadas, paisajes marinos, playas y muchos otros.

INTRODUCCIÓN

Dentro del área del tratamiento de imágenes es común encontrar problemas donde se requiere generar paisajes en los que se incluyan montañas, árboles, nubes, ríos y muchos otros elementos de la naturaleza, por lo que, desde hace algún tiempo y en particular desde el surgimiento de la teoría de Fractales se han desarrollado métodos y técnicas orientados a resolver este problema y específicamente ya se cuenta con una gran cantidad de herramientas que permiten generar montañas, nubes, plantas y otros elementos de un paisaje.

Ahora bien, comúnmente se utilizan técnicas distintas para resolver los diferentes problemas y es así que por un lado se tienen métodos para generar nubes y por otro estrellas o arboles, siendo cada método diferente y especifico.

Por ejemplo, en el caso de las plantas es común encontrar métodos en los cuales a partir de un elementos inicial y mediante un algoritmo recursivo se producen ejemplos de plantas muy naturales, por otro lado, desde los años 70's ya se encontraban aplicaciones de los Sistemas-L a la generación de plantas con muy buenos resultados.

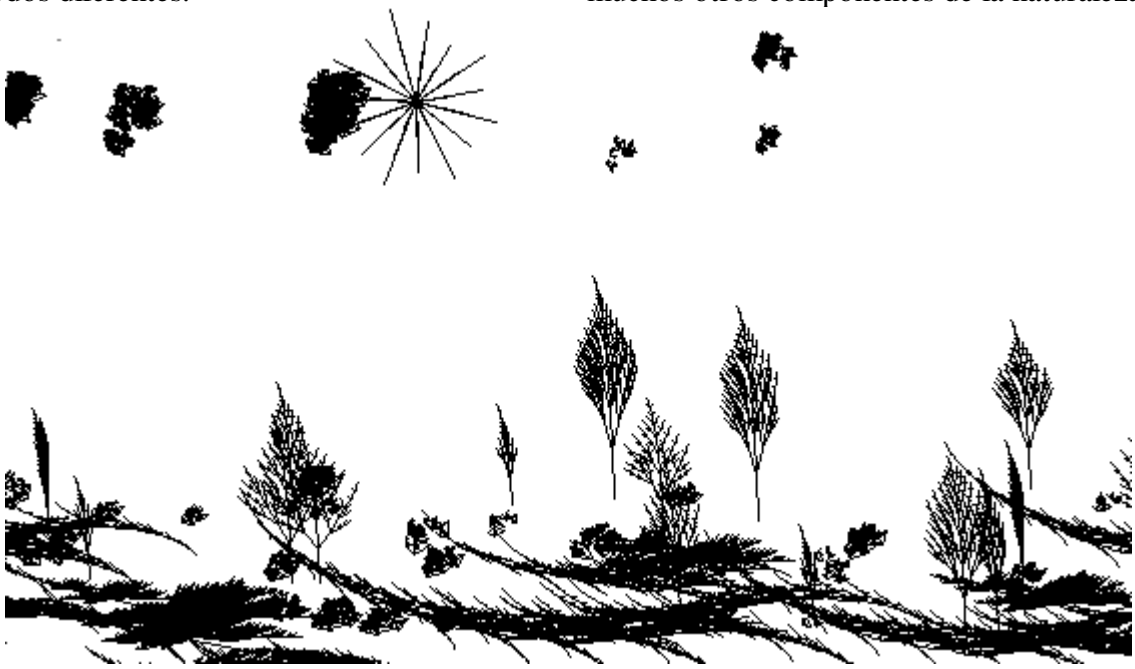
Tanto los sistemas-L como los métodos basados en un Iniciador-Generador se han aplicado a construir además de plantas también ríos, rayos y otros elementos de la naturaleza que tienen una estructura dendrítica (formada por múltiples ramas que se entrecruzan como una red neuronal o de vasos capilares).

Por otro lado, en el caso de nubes o montañas se ha tenido también un gran desarrollo y así existen métodos para generar montañas mediante particiones sucesivas de una curva o mediante el expediente de generar en 3D la imagen de un objeto y colocando en el eje Z la densidad del objeto (Por ejemplo tomar la

fotografía de una galaxia y gráficar en Z la densidad de estrellas).

Sin embargo estos métodos por lo común son independientes unos de otros y el que produce por ejemplo las montañas no se aplica para generar árboles, por lo que la construcción de un paisaje involucra el uso de múltiples métodos diferentes.

En este trabajo se presenta un herramienta basada en la Lingüística Matemática y en particular en una ampliación semántica de las Gramáticas Generativas, con la cual es relativamente fácil *representar la estructura de múltiples elementos de la naturaleza***,** por lo que prácticamente la misma rutina genera árboles, nubes, montañas, caracoles, estrellas y muchos otros componentes de la naturaleza.



1. UNA ECUACIÓN FUNDAMENTAL.

Las gramáticas generativas surgieron en 1957 a partir de los Trabajos de Noam Chomsky y desde principios de los 60's se han aplicado para representar la estructura de los múltiples lenguajes de programación que existen y poder construir los compiladores o intérpretes de estos lenguajes.

Por otro lado también desde los 60's ha sido la base de los métodos de reconocimiento sintáctico de patrones en los cuales se ve a una imagen o fenómeno que se repite cotidianamente (voz, movimiento de los planetas, jugada de ajedrez, etc.) como una 'oración' del lenguaje de imágenes o del fenómeno repetitivo, a partir de ahí se encuentra la regla gramátical o producción que representa la estructura general de este lenguaje y se asume entonces que por ejemplo, la regla gramátical de un conjunto de imágenes representa su patrón general o rasgos característicos.

Lo anterior permite desarrollar una herramienta que, a partir de la gramática de una familia de imágenes, es capaz de reconocer si una imagen en particular pertenece a la familia o no (como un compilador es capaz de reconocer si una oración pertenece o no a un lenguaje dado). Por otro lado, si se invierte el orden de entradas y salidas se puede tener un sistema capaz de general imágenes específicas a partir de una gramática dada.

Un enfoque particular de lo anterior se presentó durante los 80's, al aplicar la Lingüística Matemática al reconocimiento y generación de componentes de la naturaleza como árboles, montañas y nubes con el fin de construir Sistemas Evolutivos de la Naturaleza (en donde un sistema evolutivo es un mecanismo capaz de encontrar, construir y mantener actualizada una representación del medio que lo rodea y usar esa representación para mantenerse y resolver problemas dentro de ese medio).

Específicamente el propósito consistió en lograr que un Sistema Evolutivo tomara la imagen de una familia de objeto de la naturaleza (árbol, montaña, etc.) encontrará la regla gramátical o producción que representaba a la familia de objetos y fuera capaz de reconocer si un nuevo objeto pertenece o no a la familia. Como otro resultado además se encontró que esta era una forma muy simple de compactar objetos y de generar nuevas imágenes de la naturaleza.

En un principio las producciones encontradas eran diferentes unas de otras pero conforme se siguió atacando el problema se detectó que en muchos casos la regla gramátical encontrada era un caso particular de una regla mas general.

En principio esto no fue tan extraño, ya que, por ejemplo se detectó que diferentes tipos de arboles generaban reglas gramáticales diferentes, pero que se podían agrupar en una regla gramátical mas general, por otro lado con las montañas sucedía algo parecido y otro tanto con las nubes, por lo que al final se tenia una producción general para arboles (para estructuras dendriticas) otra para montañas y otra para nubes.

La sorpresa surgió cuando se detecto que a su vez todas esas producciones generales eran un caso particular de una regla gramátical mas general aun, que las englobaba a todas.

Encontrándose que prácticamente la estructura gramátical de cualquier elemento de la naturaleza se puede ver como un caso particular de una regla general a la que llamamos *Ecuación Fundamental* y es de la forma:

S --> e* S*

Donde **S** significa Sistema, **e** es cualquier elemento del sistema (una rama un tronco, una ladera etc.), **e*** significa que se pueden tener tantos elementos como se quieran y **S*** indica que el sistema se puede llamar tantas veces como se quiera. Observe que estamos utilizando el signo * como un factor de repetición, lo cual no es una notación común dentro de las reglas de producción, pero en este caso facilita la representación.

Ejemplos particulares de la ecuación fundamental son:

S --> e
S --> e S
S --> e SS
S --> e SS ... S
S --> e SeSe... S

2. ARBOLES, ESTRELLAS Y CARACOLES.

En particular la ecuación

S --> e

que se lee como **el Sistema S llama a e**

representa la estructura de un programa que llama a un elemento, por ejemplo la rutina que gráfica un tronco:

```
Sistema
{
 tronco
}
```

o mas especificamente

```
Sistema (x0,y0,long,w)
{
 tronco(x0,y0,long,w)
}
```

donde $\mathbf{x_0,y_0}$ representan el punto inicial del tronco, **long** su tamaño y **w** el angulo con el que crece.

Como otro ejemplo podemos ver que la ecuación

S --> e S

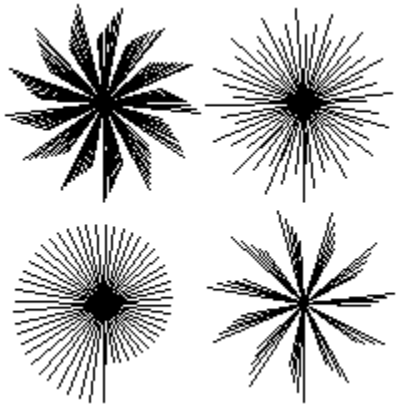
es equivalente a un sistema que genera un elemento y se llama recursivamente, con lo que, vuelve a generar otro elemento y así sucesivamente.

```
Sistema
{
  elemento
  sistema
}
```

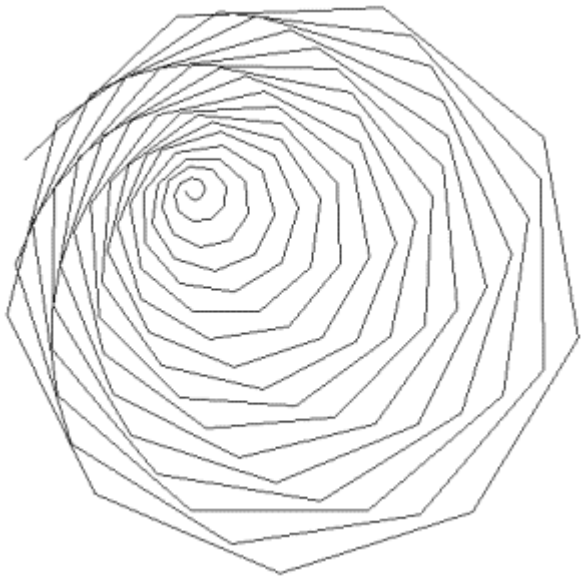
Esta ecuación engloba por ejemplo a la familia de las estrellas ya que la rutina

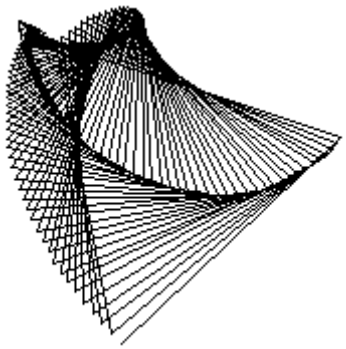
```
Sistema(x0,y0,long,w)
{
  elemento(x0,y0,long,w)
  sistema(x0,y0,long,w+w1)
}
```

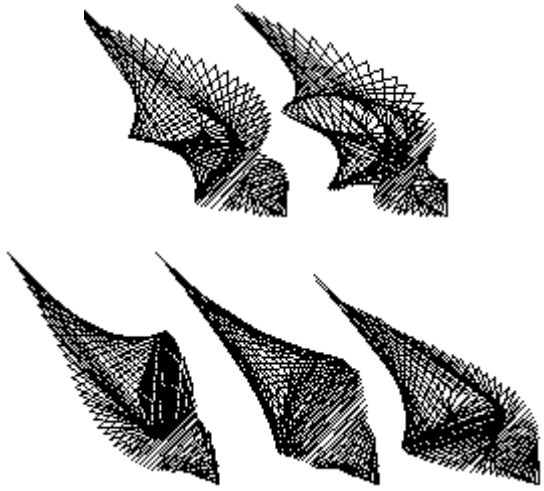
Gráfica una recta a partir de un punto (x_0, y_0), con un ángulo **w** y tamaño t y el ángulo se cambia entre llamadas; por lo que al final la rutina genera una estrella.



Por otro lado y con un cambio mínimo se genera la familia de los caracoles, para lo cual, únicamente se necesita que el punto inicial (x, y) de la nueva recta sea el punto final de la recta anterior.







Es importante observar que en el caso de un sistema recursivo el proceso continua indefinidamente, por lo que el llamado recursivo únicamente representa la posibilidad de que surja una nueva rama o el sistema siga creciendo, sin embargo si esto se genera en una computadora es importante introducir algún mecanismo que permita detener el proceso como por ejemplo una tecla de Escape, o por otro lado detener el proceso cuando se cumpla alguna condición explícita como por ejemplo que se cumplió un numero prefijado de llamados.

En muchos casos es preferible una estructura iterativa en lugar de la recursiva, sin embargo existen problemas en los cuales la representación más simple es mediante procesos recursivos.

En el caso de rutinas que dibujan árboles, la estructura iterativa ya no es funcional y en cambio la estructura recursiva es muy simple, por ejemplo la ecuación:

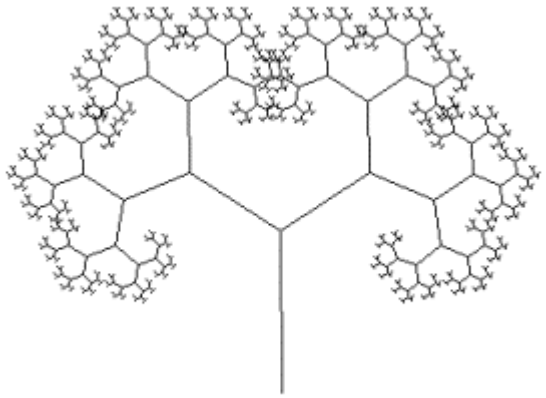
S --> e S S

representa una familia de arboles de 2 ramas

y el programa es de la forma

```
main()
{
 árbol(x0, y0,long, ángulo, nivel);
}

árbol(x,y,long,wg,nivel)                    /*S*/
{
  if (nivel> 0) /* condición de terminación*/
  {
    rama(x,y,long,wg,x1,y1);                /*e*/
    árbol(x1,y1,long/lon1,wg+w1,nivel-1);   /*S */
    árbol(x1, y1, long/lon2,wg + w2, nivel -1); /*S*/
  }
}
```



Donde lon_1, lon_2, w_1 y w_2 son parámetros que indican los cambios de tamaño y ángulo de las ramas.

Ahora bien, si se quiere dibujar un árbol con tres ramas, simplemente se pondría una llamada recursiva más y así sucesivamente, construir árboles con tres, cuatro, cinco o más ramas se vuelve trivial, ya que su estructura queda:

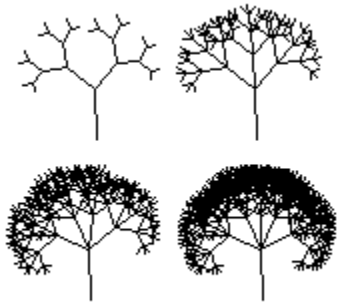
S --> e SSS
S --> e SSSS
..

S --> e SS.....S

y *en general la estructura de cualquier árbol es de la forma*

S --> e S*

y los programas son muy simples.



3. UN PROGRAMA GENERALIZADO.

Construir un programa para cada tipo de árbol puede llegar a ser tedioso porque al final todos los programas son prácticamente idénticos y es más fácil pensar en un programa general al cual simplemente se le indique cuantas ramas se quieren y se llame a la rutina tantas veces como se necesite.

Por ejemplo, la rutina

```
árbol(x,y,long,wg,nivel)
{
 if (nivel>0)
 {
  rama(x,y,long,wg,xl,yl)
  ind=1
  while  ind <= n
    árbol(x1,y1,long/lon[ind],wg+w[ind++],nivel--)
 }
}
```

representa precisamente a la ecuación

S --> e S*

Y dibuja árboles con n ramas (donde n es una variable que representa el número de ramas).

Otra forma alternativa de representación sería que en lugar de pasarle el número n de llamadas a la rutina, se le pasará una lista de la forma eSeeSSSeS con tantas e's y S's como se quiera.

La ventaja de este enfoque es que el sistema se puede generalizar para representar llamados a árboles 'raros' como por ejemplo

S -->eSe

S -->eSSe

S -->eeSeSS

S -->eSSeeSe

y cualquier combinación de elementos y llamadas, para lo cual únicamente se necesita preguntar si el comando es una **e** o una **S** como se ve a continuación:

```
gramática[]= "eSeeSS"
main()
{
 árbol(x0,y0,l0,w0,nivel)
}
árbol(x,y,long,wg,nivel)
{
 si (nivel>0)
 {
  i=l
  mientras (gramática[i] != fin de renglón)
   {
     si (gramática[i]='e') linea(x,y,l,w,x1,y1)
     si (gramática[i]='S')
           árbol(x1,y1,long/lon[i],wg+w[i],nivel--)
     i++
   }
 }
}
```

El anterior es un Sistema Generador de Arboles y representa a la estructura

S --> e*S*

Si en el arreglo llamado gramática se almacena:

S --> e
El sistema genera una línea.

S --> eS
El sistema genera caracoles o estrellas

S --> eSS
El sistema genera árboles con dos ramas

S --> eSSS
El sistema genera árboles con tres ramas

y así sucesivamente.

De donde resulta que el mismo programa que gráfica líneas, gráfica caracoles y arboles y todos son un caso particular de la estructura

S --> e* S*

Al anterior sistema se le pueden hacer múltiples modificaciones como:

- Sustituir la rutina línea por una rutina que genere troncos de diferentes anchos, largos, colores, texturas, en forma de arco, etc.

- Parametrizar el máximo de elementos como li, wi. y pasar diferentes valores para cada caso o aún más introducir secuencias aleatorias basadas en la distribución estadística de los elementos de una planta y en sus probabilidades de crecimiento, muerte o reproducción.
- Introducir más de un tipo de elemento y generar con líneas, arcos, troncos círculos, triángulos, esferas u otros elementos.

- Introducir múltiples estructuras y permitir que unas se llamen a otras, por ejemplo, tener la estructura de árbol, hoja y flor y que en cierto momento árbol llame a hoja o flor.

Con lo que un programa de este tipo es capaz de generar plantas de múltiples tipos únicamente cambiando la gramática que representa la estructura de la planta. Y para lograr lo anterior simplemente se requiere generalizar el programa para que pueda llamar al i-esimo elemento (tronco, línea, arco, etc.) y a la i-esima estructura o producción de la gramática.

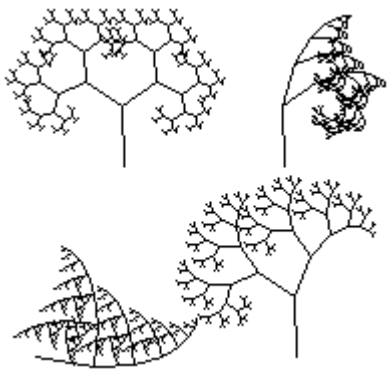
En general la estructura de cualquiera de los sistemas anteriores es de la forma

Si --> e_j* S_k*

donde $\mathbf{e_j}$ es cualquier elemento del sistema y $\mathbf{S_k}$ cualquier producción de la gramática que representa al sistema.

Lo anterior nos proporciona una *herramienta generalizada para el desarrollo de sistemas* y en particular de árboles *a partir de la Lingüística Matemática*, ya que únicamente es necesario encontrar la gramática del sistema que se quiera producir y proporcionárselo al

generador para que este obtenga el sistema específico con la estructura que se desee.



4. ARBOLES, NUBES, MONTAÑAS Y PAISAJES.

Sin embargo el asunto no queda ahí ya que no se ha dicho prácticamente nada de los parámetros del sistema (En este caso x,y,l,w).
Al empezar a jugar con los parámetros el efecto que se obtiene es precioso, ya que, cada uno de ellos al ser modificado puede ocasionar cambios radicales en la forma de los objetos generados, con lo que, dos objetos con la misma estructura interna (gramática) pueden tener una forma radicalmente diferente.

Por ejemplo si tomamos el siguiente caso particular

```
main()
{
 w0=c0; w1=c1; w2=c2; w3=C3; ind=cd;
 árbol (x0,y0,l,w0,ind)
}
árbol(x1, y1, l, w, ind)
{
  si ind>0
 {
    linea(x,y,l,w,x1,y1)
    árbol(x1,y1,l/1.2,w+w1,ind-1)
    árbol(x1,y1,l/1.55,w+w2,ind-1)
    árbol(x1,y1,l/1.8,w+w3,ind-1)
 }
}
```

donde w_0, w_1, w_2, w_3 son el ángulo inicial y los incrementos de ángulos para cada llamada recursiva.

Podemos encontrar que si fijamos

w_0=-72 w_2=72 w_3=144

y modificamos w_1 entre 1 y 360 grados podemos generar una familia completamente diferente, ya que si por ejemplo

- w_1=24 tenemos un bosque de helechos
- w_1=4 tenemos una nube en forma de perro
- w_1=139 tenemos una nube
- w_1=169 tenemos un fractal de dragón

y así sucesivamente hasta regresar a los helechos.

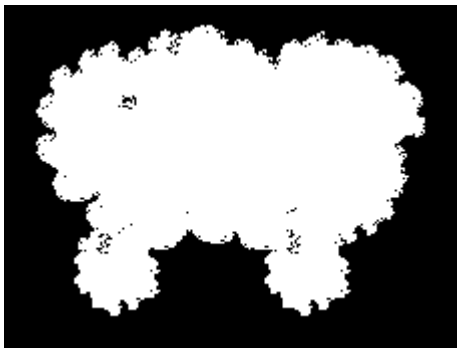
Si el cambio del ángulo se realiza con saltos muy pequeños se puede observar como un elemento se transforma en otro , con lo que para generar un bosque de helechos o una nube solo es necesario cambiar el valor de un solo parámetro de todo un sistema.

El anterior efecto se produce si mantenemos fijos w_0, w_2, w_3 y modificamos w_1, ahora bien si modificamos cualquiera de los 4 ángulos la cantidad de elementos que se pueden generar es enorme e incluye nubes, arboles y montañas entre otros.

En general las ecuaciones de la forma

S --> eSS ... S

permiten generar múltiples familias de arboles, montañas, nubes, ríos y prácticamente cualquier elemento de la naturaleza.



Por ejemplo si :

- Fijamos w_0=-27 w_1=6 w_2=172
  y modificamos w_3 se genera una familia de laderas de montaña.

- Fijamos w_0=-7 w_1=6 w_2=72
  modificamos w_3 se genera una familia de arboles en una ladera y arboles con reflejo.

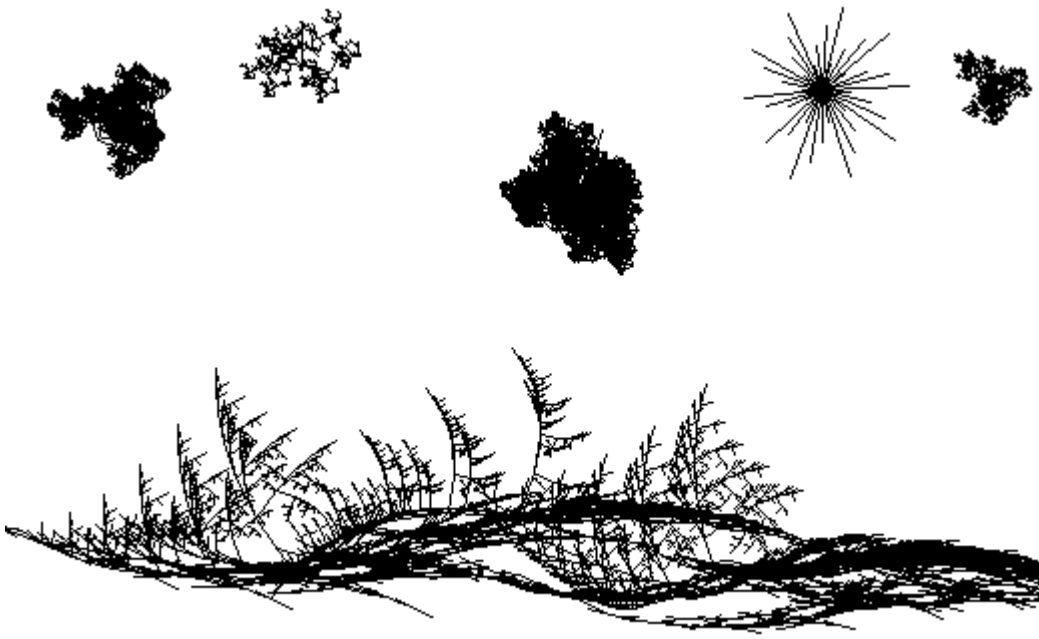
- Fijamos w_0=-72 w_2=172 w_3=144
  y modificamos w_1 se genera una familia de alacranes

Como se puede ver, con una rutina de unos cuantos renglones y cambiando solo los ángulos se tiene prácticamente todos los elementos de un paisaje, por lo que, un programa que genera un paisaje se limita a un conjunto de llamados a la misma rutina, a la cual le pasan solo las modificaciones de los parámetros, como se puede ver en el siguiente ejemplo:

```
int w0,w1,w2,w3,color;
main()
{
  /*estrella */color=3;
   wl=37; estrella(450,90,1,27,47);
  /*nube */ color=2;
  w1=238; w2=273; w3=144; paisaje(100,80,37,-72,9);
  w1=188; w2=243; w3=144; paisaje(340,110,59,-
72,9);
  w1=238; w2=273; w3=125; paisaje(S50,80,23,-23,7);
  w1=287; w2=103; w3=144; paisaje(l90,60,19,-92,9);
  /*montaña*/
  color=3;w1=6;w2=172;w3=186;
          paisaje(425,350,63,173,7);
  color=5;w1=6;w2=172;w3=1;
          paisaje(210,320,63,169,7);
  color=9;w1=6;w2=172;w3=186;
          paisaje(280,325,63,8,7);
  color=6;w1=6;w2=172;w3=1;w0=172;
          paisaje(380,335,63,w0,7);
  color=6;w1=6;w2=172;w3=1;w0=173;
          paisaje(380,335,63,w0,7);
  /*árbol */ color=l;
    w1=8; w2=72; w3=17;
           paisaje(225,315,43,-17,7);
   w1=-12; w2=352; w3=97;
          paisaje(340,315,43,15,7):
  w1=8; w2=72; w3=351;
           paisaje(443,340,43,0,7);
}

paisaje(int x0, int y0, int l, int an, int ind)
{
 int x1,y1;
 if  (Ind >0)
 {
   setcolor(ind+color);
   recta(x0,y0,l,an,&x1,&y1);
   paisaje(x1 ,y1,l/1 .2,an+w1,ind-1);
   paisaje(x1,y1,l/1.55,an+w2,ind-1);
   paisaje(x1 ,y1,l/1.8,an+w3,ind--);
 }
}

estrella(int x0,int y0,int l,int an,int ind)
{
  int x1,y1;
  if  (Ind > 0)
  {
   setcolor(ind+color);
   recta(x0,y0,l,an,&x1,&y1);
   estrella(x0,y0,l+1,an+w1,ind--);
  }
}
```



CONCLUSIÓN.

En este documento se presento una aplicación de la Lingüística Matemática a la generación de paisajes, para lo cual: en primer lugar se planteo que existe una ecuación fundamental de la naturaleza de la forma

S--> e*S*

que engloba a las ecuaciones que representan la estructura de troncos, caracoles, estrellas, arboles, nubes y montañas, entre otros. Lo anterior es fundamental ya que nos muestra que múltiples fenómenos de la naturaleza tienen la misma estructura.

Mas adelante se vio como a partir de la estructura Lingüística se obtiene directamente un programa y en particular se presento un sistema basado en la ecuación fundamental y se mostró como cambiándole simplemente algunos parámetros genera una gran cantidad de elementos.

Este enfoque generaliza y engloba dentro de la lingüística matemática múltiples herramientas como los métodos iniciador-generador y los de los sistemas-L, con lo que, se puede aprovechar todo el poderío de esta herramienta que se utiliza desde hace muchos años para construir compiladores, reconocer formas, medir la complejidad de sistemas y en general para encontrar y representar la estructura de múltiples problemas de la Informática, por lo que este documento es también una invitación al uso de la Lingüística Matemática dentro de la graficación y animación.

Como se puede ver estamos llegando a que la estructura de una cantidad enorme de elementos de la naturaleza es la misma, ya que, por ejemplo únicamente cambiando algunos ángulos dentro de una rutina se pueden generar nubes, laderas de montañas, arboles, alacranes, hojas y paisajes en generar.

AGRADECIMIENTOS.

Muchas gracias a Marina Vicario Solorzano y Francisco Aguilar Vallejo de TECCIZ de México, Cuitlahuac Cantu de ORSA, Sergio Rivera, Francisco Molina C. y Cruz Luna Reyes de UPIICSA-IPN por su ayuda y contribución a este trabajo.

REFERENCIAS.

1. **Kenton** Musgrave, Craig E. **KoIb** y Robert S. **Mace**. *The Synthesys and Rendering of Eroded Fractal terrains.* In Computer Graphics, Vol 23(3), julio 1989.

2. Przemyslaw **Prosinkiewicz**, Aristid **Lindenmayer** y James **Hanan**. *Developmental Models of Herbaceous Plants for Computer Imagery Purposes*. Computer Graphics, Vol 22(4), Agosto 1988

3. Sofia **Bueno Peralta** y Antonio **Simancas López**. *Generador de Arboles Fractales.* en Memorias del III Congreso Nacional sobre Informática y Computación, Jalapa, Ver. México, Octubre 1990.

4. Peter E. **Oppenheimer**. *Real Time Design and Animation of Fractal Plants and Trees*. In Computer Graphics, Vol 20(4) SIGGRAPH'86, August 1986.

5. Heinz-Otto **Peitgen** and Dietmar **Saupe** (Editores). *The Science of Fractal Images*. Ed. Springer-Verlag, 1988.

6. Fernando **Galindo Soria**. *Sistemas Evolutivos: Nuevo Paradigma de la Informática.* en Memorias XVII Conferencia Latinoamericana de Informática, Caracas Venezuela, julio de 1991.

7. Fernando **Galindo Soria**. *Sistemas Evolutivos*. en Boletín de Política Informática. México, Septiembre de 1986.

8. Rafael C. **Gonzalez** y Michael C. **Thomason**. *Sintactic Pattern Recognition*. Ed. Addison-Wesley.

9. Fernando **Galindo Soria.** *Aplicaciones de la Lingüística Matemática y los Fractales a la Generación de Imágenes.* en Memorias Simposium Nacional de Computación. México, Nov. de 1991.

10. Emmon **Bach.** *Teoría Sintáctica.* Ed. Anagrama.

11. Salomaa. *Formal Languages*. Ed. Academic Press.

12. Herbert. A. **Simon.** *Las Ciencias de lo Artificial*. Ed. ATE

13. Noam **Chomsky**. *Estructuras Sintácticas.* Ed. Siglo XXI

14. Hopcroft y **Ullman**. *Formal Languages and Their Relation to Automata.* Ed. Addison-Wesley