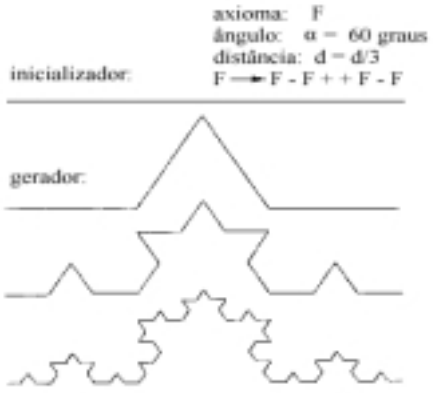
CAPÍTULO 6

Modelos Gráficos por L-Systems

Os **Lindenmayer-systems**, ou abreviadamente **L-Systems**, foram imaginados inicialmente como uma teoria matemática para o desenvolvimento de plantas. Inicialmente, não enfatizando a modelagem de plantas complexas, mas a topologia dos vegetais e as relações de vizinhança entre as células e módulos. Depois, diversas interpretações geométricas foram incorporadas para que se tornasse uma ferramenta versátil na modelagem de plantas.

O conceito central dos L-Systems é a **busca e substituição**. Objetos complexos são definidos por sucessivas **substituições** em objetos inicialmente simples.

A forma de construção da "Snow Flake curve" (proposta por Koch em 1905) serve como um bom exemplo desta técnica. Começa-se com duas formas; um **iniciador** e um **gerador**. O gerador é uma linha quebrada feita de N lados iguais de comprimento r. Em cada estágio da construção a linha quebrada tem suas partes retas substituidas por cópias do gerador, deslocado e reduzido de modo a ter os mesmos pontos limites da reta a ser substituida.



Uma das formas mais estudadas de sistemas de busca e substituição é nas operações com strings de caracteres. Chomsky[1] aplicou sistemas deste tipo no estudo da sintaxe de linguagens naturais. Estas formas foram usadas por Backus[2] and Naur[3] para definições formais da linguagem de programação ALGOL. Em 1968, o biólogo A. Lindenmayer introduziu um método essencialmente diferente destas aplicações em sintaxe e gramática. Ao invés de aplicações apenas sequênciais, como as gramáticas de Chomsky, os L-systems aplicam também substituições paralelas. Esta diferença é motivada pelas aplicações à biologia dos L-systems.

6.1 - Gramáticas

Vejamos intuitivamente o que são **gramáticas**: considere uma palavra inicial (**axioma**), **regras de substituição** associadas a cada letra (rewriting rules ou production rules) e um **número de cíclos** de repetição do processo.

Por exemplo se as **regras de substituição** forem: $a \longrightarrow ba$
$b \longrightarrow an$
$n \longrightarrow b$

Supondo que a palavra inicial (**axioma**) seja b, a derivação de um L-systems de **5 ciclos** seria :

b
bab
anbaan
babanbabab
anbaanbabanbaanbaan

6.2 - Construções Gráficas

Para modelagens complexas, interpretações gráficas mais sofisticadas são necessárias. Aspectos geométricos, como linhas e ângulos, devem ser considerados. Uma forma básica concentra-se na idéia da **turtle interpretation**: onde a posição corrente (**turtle's position**) é representada pelas coordenadas (x, y) e um ângulo (α), chamado de **heading** indica a direção que estará voltado o próximo caminhamento. A partir de dado **tamanho de passo**, **d** (step size), e **incremento de ângulo**, δ, uma sequência de comandos para a **turtle** seria representada pelos símbolos:

F - move para a frente um passo de comprimento **d**. Assim se a turtle estava posicionada em (x_0, y_0, α) passará para (x, y, α) onde $x = x_0 +$ d $cos\alpha; y = y_0 +$ d $sen\alpha$. E a linha entre (x, y) e (x_0, y_0) é desenhada.

f - move a turtle para (x, y) sem desenhar a linha.

$+$ - gira a turtle de um ângulo δ para a esquerda. A próxima posição será $(x, y, \alpha + \delta)$ pois considera-se positivo ângulos no sentido anti-horário.

$-$ - gira a turtle para a direta. A próxima posição será $(x, y, \alpha - \delta)$.

Assim dada uma string inicial, l, um estado inicial (x_0, y_0, α_0) e parâmetros **d** e δ, a turtle interpretation de uma string é a figura gerada pela resposta aos comandos determinados pelas substituições.

Figuras complexas podem ser obtidas pela associação das regras de substituição e interpretação de strings. Por exemplo a fractal "quadratic Koch island" é gerada pelas strings e regra de substituição (`Como são os primeiros passos desta fractal?`) :

$\delta = 90^o$

$l : F - F - F - F$

d : d/4

$p : F \longrightarrow F - F + F + FF - F - F + F$

O "arquipelago de Koch", que segue, ilustra a forma como os L-systems são aplicados para codificar construções usando f e duas regras de substituição (`Como são os primeiros passos desta fractal?`).

$\delta = 90^o$

$\alpha : F + F + F + F$

d: d/6

$F \longrightarrow F + f - FF + F + FF + Ff + FF - f + FF - F - FF - Ff - FFF$

$f \longrightarrow ffff$

6.3 - Descrição formal dos L-systems

Usando a notação: **V** - para representar um alfabeto; **V*** - para representar o conjunto de todas as palavras sobre **V**; w $\in$ **V** - para uma palavra inicial associada, não vazia, chamada axioma; e **P** - para representar um conjunto de regras de produção no espaço **V** $\times$ **V***, um L-system será então representado pela trinca $<$ **V , w , P** $>$.

As produções serão representadas pelos pares (**c , s**): para cada letra **c** $\in$ **V** existe pelo menos uma palavra **s** $\in$ **V*** tal que **c** $\longrightarrow$ **s**.

O L-system será determinítico se, e só se, para cada $\mathbf{c} \in \mathbf{V}$ existir um único $\mathbf{s} \in \mathbf{V^*}$ tal que $\mathbf{c} \longrightarrow \mathbf{s}$.

O alfabeto **V** consiste de todos os símbolos que aparecerem no axioma e nas regras de produção. Se alguma regra de produção não for especificada ela será assumida como **identidade** (Exemplo: $+ \longrightarrow +$; $- \longrightarrow -$).

A grande vantagem dos L-system é a descrição concisa dos objetos baseada na simulação do desenvolvimento. Uma forma de modelar um processo consiste basicamente em **"capturar seu desenvolvimento"**. Este modelo matemático pode ser usado para gerar imagens biológicamente corretas de plantas em diferentes idades e criar sequências de imagens ou animações que representam seu crescimento no tempo. O modelo também registra as relações espaço-tempo (como envelhecimento ou alterações de estações) entre partes de plantas. Pois em muitos casos órgãos em vários estágios de desenvolvimento são observados ao mesmo tempo (por exemplo algumas flores podem estar em botão, outras desabrochadas e algumas se desenvolvendo para gerar frutos).

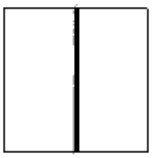
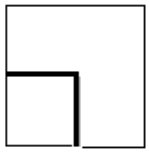
Geralmente deseja-se construir L-systems que capturam um dado processo de desenvolvimento o que é chamado **inference problem**. Alguns algorítmos de construção de plantas e figuras já estão disponíveis (por exemplo para Java, no JDK há exemplo de um programa para construção da curva de Koch, que pode ser rapidamente alterado para construir qualquer uma das fractais deste capitulo).

No entanto, o método de desenvolver novas estruturas (regras de substituição que resultarão determinada figura) ainda é bastante intuitivo. Basicamente dois modos de operações dos L-system são mais usados para "Turtle interpretation". Eles são chamadas de "edge rewriting" e "node rewriting". No primeiro método substitui-se figuras por lados de polígonos e no segundo as operações são feitas nas vértices de polígonos. Ambos baseando-se na captura da estrutura recursiva da figura.

- Edge Rewriting - Este método pode ser entendido como uma extensão da constrúção de Koch já descrita: cada lado é substituido por uma sequência de operações descritas pelas regras representadas.

- Node Rewriting - Neste caso os nós (regiões do espaço ou do plano) são substituidos por sequências de operações.

Como o primeiro já deve estar bem claro, vamos exclarecer melhor este segundo método. Para uma figura no plano, um nó será uma região do espaço. Um quadrado pode ser considerado como a região do espaço onde a figura será desenhada. Em cada iteração esta área inicial será substituida por um "quadriculado" de tamanho cada ver menor. As regras de substituição indicam como tratar os nós a cada iteração. Os demais elementos têm o mesmo sentido do método anterior.

Exemplificando: suponha $R_0 =$  , $L_0 =$  ,

as regras de substituição:

$L_{n+1} = + R_nF - L_nFL_n - FR_n +$

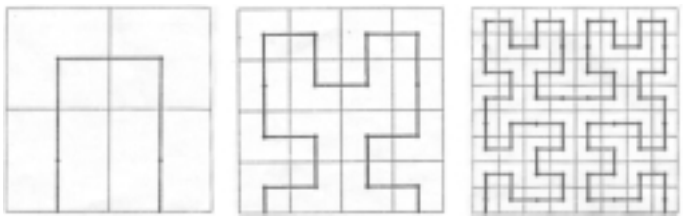
$R_{n+1} = -L_nF + R_nFR_n + FL_n -$

onde F, + e - têm o significado já anteriormente descrito e $\delta = 90^o$. Uma forma de avaliar cada string L_n ou $I\!R_n$ para n > 0 é avaliar strings sucessivas em ordens decrescente dos

índices. Para se avaliar, por exemplo L_2, usa-se a forma recursiva para n = 2, e depois para n = 1, até só serem vistos strings em L_0 e R_0. Assim L_2 será:

$$L_2 = +R_1F - L_1FL_1 - FR_1+$$

$$\begin{aligned} L_2 =& (-L_0F + R_0FR_0 + FL_0-)F - (+R_0F - L_0FL_0 - FR_0+) \\ & F(+R_0L - L_0FL_0 - FR_0+) - F(-L_0F + R_0FR_0 + FL_0-)+ \end{aligned}$$



As duas formas, edge ou mode rewriting, são equivalentes. Na prática a questão de qual será usada para descrever um processo é apenas questão de conveniência.

6.4 - Estruturas Ramificadas

Árvores têm lados que são identificados e direcionados. Os lados em sequência formam caminhos a partir de um nó raiz até um nó terminal que são denominados ramos. Segmentos que possuem pelo menos um outro segmento seguinte são chamados **internode**. Um segmento terminal é chamado de **apex**. Uma **árvore axial** é um tipo especial de árvore na qual em cada nó, pelo menos um segmento de reta se origina. Uma sequência de segmentos de uma árvore é chamada de eixo se começa na raiz (segmento de ordem zero) e tem segmento até a ordem (n+1).

Para modelar estruturas ramificadas existem rewriting rules especiais chamadas: **tree production**, onde são usadas extensões com colchetes. Estes colchetes são interpretados como:

[- armazenar (PUSH) o estado corrente (posição, angulo de orientação) da turtle na pilha.
] - apanhar na pilha o estado corrente (POP) e produzi-lo na turtle.

Em ambos não são desenhadas linhas, embora a posição da turtle seja alterada.

Exemplo: F [+ F] [- F [- F] F] F [+ F] [- F], $\delta = 45^o$ (`que resultá está string?`)

Estruturas de árvores também podem ser usados em formas recursivas, como a seguinte: (`que tal desenhá-las!`)

$$\begin{aligned} & n = 2 \quad \delta = 25,7^o \\ & F_0 \longrightarrow F \\ & F_1 \longrightarrow F_0[+F_0]F_0[-F_0]F_0 \\ & F_2 \longrightarrow F_1[+F_1]F_1[-F_1]F_1 \\ F_2 =& (F_0[+F_0]F_0[-F_0]F_0)[+(F_0[+F_0]F_0[-F_0]F_0)] \\ & (F_0[+F_0]F_0[-F_0]F_0)[-(F_0[+F_0]F_0[-F_0]F_0)](F_0[+F_0]F_0[-F_0]F_0) \end{aligned}$$

Outro exemplo (`para você desenhar`) agora com 2 funções, $n = 2$, $\delta = 20^o$, sendo:

$$
\begin{aligned}
x_0 &\longrightarrow \spadesuit \\
x_n &\longrightarrow F_n[+x_{n-1}]F_n[-x_{n-1}] + x_{n-1} \\
F_n &\longrightarrow F_{n-1}F_{n-1} \\
x_2 &= F_2[+x_1]F_2[-x_1] + x_1 \\
&= F_1F_1[+x_1]F_1F_1[-x_1] + x_1 \\
&= F_1F_1[+F_1[+x_0]F_1[-x_0] + x_0]F_1F_1[-F_1[+x_0]F_1[-x_0] + x_0] + F_1[+x_0]F_1[-x_0] + x_0 \\
&= F_0F_0F_0F_0[+F_0F_0[+x_0]F_0F_0[-x_0] + x_0] \\
&\quad F_0F_0F_0F_0[-F_0F_0[+x_0]F_0F_0[-x_0] + x_0] + F_0F_0[+x_0]F_0F_0[-x_0] + x_1
\end{aligned}
$$

6.5 - Sistemas Randômicos

Cada L-system determinístico irá produzir a mesma planta. Quando diversas forem combinadas no mesmo quadro o resultado será "meio artificial". Neste caso alguma irregularidade deve ser artificialmente induzida ao processo. **Efeitos randômicos** podem ser introduzidos nos L-systems ou nas "turtle interpretations" (ou em ambos) introduzindo mais de uma regra de substituição a cada simbologia, cada uma com uma probabilidade própria descrita entre parênteses. Como exemplo de L-systems estocásticos temos:

$$
\begin{aligned}
&\text{F} \rightharpoonup (1/3)F[+F]F[-F]F \\
&\text{F} \rightharpoonup (1/3)F[+F]F \\
&\text{F} \rightharpoonup (1/3)F[-F]F
\end{aligned}
$$

Assim se $\delta = 45^o$ e $n = 2$ poderemos ter entre outros resultados:

$$
\begin{aligned}
F_2 &= F_1[+F_1]F_1[-F_1]F_1 = F_0[+F_0]F_0[-F_0]F_0[+F_0[+F_0]F_0[-F_0]F_0] \\
&\quad F_0[+F_0]F_0[-F_0]F_0[-F_0[+F_0]F_0[-F_0]F_0]F_0[+F_0]F_0[-F_0]F_0[-F_0]F_0
\end{aligned}
$$

ou

$$
F_2 = F_0[+F_0]F_0[+F_0[-F_0]F_0]F_0[+F_0]F_0[-F_0[+F_0]F_0]F_0[+F_0]F_0
$$

ou

$$
F_2 = F_1[+F_1]F_1 = F_0[+F_0]F_0[-F_0]F_0[+F_0[+F_0]F_0[-F_0]F_0]F_0[+F_0]F_0[-F_0]F_0
$$

6.6 - L-systems sensíveis ao contexto

Até agora as produções vistas foram independentes do contexto em que as formas predecessores e sucessoras apareciam. No entanto pode-se propor uma produção dependente do contexto em que aparecem.

Exemplo de produções sensíveis ao constexto são :

$$
a_l < a > a_r \longrightarrow X
$$

o que se quer dizer com os símbolos $<$ e $>$ é que a letra a produzirá o processo X apenas se for precedida de a_l e sucedida de a_r . Assim a_l irá fomar o contexto esquerdo (left) de a e a_r o contexto direito de a (right).

`Entendeu? então o que produzirá?` : Se $F_a =|$ e $F_b =\|$

$$F \longrightarrow F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a$$

$$F_b < F_a \longrightarrow F_b$$

O símbolo "# ignore : " pode ser usada para ignorar alguns símbolos neste processo de produção. Por exemplo **# ignore : -** indica que se ocorrer: $F_b < F_a$ ou $F_b- < F_a$ no processo o efeito é idêntico.

`O que produzirá então as linhas que seguem?`

$$\#ignore: \quad + \quad --$$

$$F \longrightarrow F_b[+F_a]F_a[-F_a]F_a[+F_a]F_a$$

$$F_b < F_a \longrightarrow F_b$$

6.7 - Modelagem em 3D

O principal conceito para extender os L-systems para 3D é a representação da posição corrente por 3 vetores: $\vec{H}$, $\vec{L}$, $\vec{U}$ indicando a posição da cabeça, a sua direção à esquerda (left) e sua direção para cima (up).

Estes vetores são unitários e ortogonais (de modo que $\vec{H} \times \vec{L} = \vec{U}$). Isso é, pode-se supor que pela tartaruga passem um sistema de eixos ortogonais, com centro no (centro do) seu casco, e com direções tais que $\vec{H}$ aponte para sua cabeça, $\vec{L}$ para seu lado esquerdo e $\vec{U}$ para cima. O plano em que ela repousa é o plano dos vetores $\vec{H}\,\vec{L}$, de modo que os ângulos no plano sejam ângulos em tormo do eixo $\vec{U}$. Girá-la em trono de $\vec{L}$, corresponde a fazê-la subir ou descer.

A rotação da turtle de um algulo α é expressa pela equação:

$$[\ \vec{H}^1\vec{L}^1\vec{U}^1\ ] = [\vec{H}\ \vec{L}\ \vec{U}\ ][R\ ]$$

onde R é uma matriz de rotação (3×3). Se α indica uma rotação de determinado ângulo em torno dos vetores $\vec{U}, \vec{L}, \vec{H}$, uma rotação em torno de $\vec{U}$ é representado por :

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

para uma rotação em torno de $\vec{L}$ a matriz de rotação é:

$$R_L(\alpha) = \begin{vmatrix} cos\alpha & 0 & -sin\alpha \\ 0 & 1 & 0 \\ sen\alpha & 0 & cos\alpha \end{vmatrix}$$

e em torno de $\vec{H}$:

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

Assim para a orientação espacial da turtle teremos que introduzir símbolos novos para as rotações em torno dos dois outros eixos:

$+$ - giro de α à esquerda usando $R_u(+\alpha)$;
$-$ - giro de α à direita usando $R_u(-\alpha)$
$\&$ - subida do rabo de um ângulo α usando $R_L(\alpha)$
$\wedge$ - subida da cabeça de um ângulo α usando $R_L(\alpha)$
$\backslash$ - giro do casco para à esquerda de α usando $R_L(+\alpha)$
$/$ - giro do casco para à direita de α usando $R_L(-\alpha)$
$|$ - giro de 180^o usando R_U (180^o)
! - incremeta o diâmetro da linha usada para desenhar
, - incremeta o índice corrente de côr usada para desenhar

Com estes símbolos diversas figuras tridimensionais podem ser produzidas `que tal voce inventar uma!`.

Referências

[1] N. Chomsky, "Three models for the description of language", *IRE Transations on Information Theory*, 2 (3) , 113 - 124, 1956.

[2] J. W. Backus, "The syntax and semantics of the proposed international algebraic language of Zurich ACM - GAMM Conference",*In Proc. Intl. Conf. on Information Processing*, 125-132, UNESCO, 1959.

[3] P. Naur et al., "Report on the algorithmic language ALGO-60',*Communications of the ACM*, 3(5), 299-314, 1960, *Revised in Comm. ACM*, 6(1) ,1 - 17

[4] P. Prusinkiewicz, A. Lindenmayer,"The Algorithmic Beauty of Plants", Springer-Verlag, 1990.