一种通用的植物逼真几何建模方法

王莉莉 赵沁平
(北京航空航天大学计算机学院虚拟现实新技术国家教育部重点实验室,北京 100083)

摘 要 针对使用 L 系统进行植物几何建模的具体过程随规则定义的变化而变化的问题,提出了一种较为通用的基于 L 系统规则语言分析器的解决方法,即通过归纳和抽象得到可以定义多种 L 系统规则的语言 L-plants,并为其构造语言分析器,完成 L 系统开始状态和规则的识别,进行规则替换,以形成最终的字符串,最后使用形状语法对字符串进行解释,建立出植物的几何模型. 实验证明,该方法可以较大幅度地提高植物几何建模的效率.

关键词 计算机图形学(520·6030) L 系统 植物建模 语法分析器生成器 词法分析器生成器
中图法分类号: TP391.41 **文献标识码**: A **文章编号**: 1006-8961(2003)08-0932-06

A General Method for Plants Modeling

WANG Li-li, ZHAO Qin-ping
(*The Key Laboratory of Virtual Reality Technology, Ministry of Education, China, Computer School, BeiHang University, Beijing* 100083)

Abstract When graphically modeling plants, the process of L-system changes with different rule definitions. As a general method, a language parser based on L-system rules is presented in this paper. Firstly, this method acquires and constructs the parser of, through induction and abstraction, L-plants which is a language able to define various L-system rewrite rules. Secondly, the start state and the rules of L-system are recognized with the parser and then the rule replacement is executed to produce a string. At last, the shape grammar is used to model a specific plant while it parsing the string. This plant modeling method proves to be highly efficient with experiments and with it users can pay more attention to designing the shape of plants and defining the rules of L-system than to thinking about the details of how to replace in each step. This method also has good expansibility. When users want the parser to recognize other types of rules of L-system, they can simply add some new syntax expressions in syntax definition files.

Keywords Computer graphics, L-system, Plants modeling, YACC, LEX

0 引 言

植物是存在于真实场景中的常见物,如树林、草地、灌木丛、绿化带等. 由于植物是自然景物,结构复杂,并具有不规则的外形,所以在虚拟场景,尤其是在布满道路、房屋等规则的人造建筑物模型的虚拟城市场景中,加入一些精细的植物模型可以大大增强场景的真实感.

目前,在视景仿真的应用中,主要采取了以下几种建立植物几何模型的方法:十字交叉法、单个多边形的 BillBoard 法和基于 L 系统的建模方法. 十字交叉法是建立两个双面可见的矩形多边形,并使它们十字交叉,然后在每个多边形上映射经过背景透明处理的植物纹理;单个多边形的 BillBoard 法是指建立单个双面可见的矩形多边形,并使用植物的纹理进行映射,在实时绘制过程中,根据视点位置和视线的方向对该面片进行以自身中线为轴的旋转,使其法线方向始终与视线方向平行,达到较好的视觉效果. 上述两种方法建模简单,可达到视景仿真中对

基金项目:国家"八六三"高技术研究发展计划基金(2001AA115130)
收稿日期:2002-11-18;改回日期:2003-03-11

植物建模最基本的要求，但同时也存在着一定的问题，纹理映射使用的是经过处理的从某一角度拍摄的植物照片，所以保持了枝叶间在视觉上的深度层次感，但由于该模型毕竟是由一、二个多边形构成的，因此不能很好地表示植物的枝叶与虚拟场景中其他模型之间的深度层次感；当视点高于矩形多边形时，逼真感差；另外，使用上述方法建立的植物模型在应用中不能与环境进行实时交互，例如树上的枝叶不能随风摆动.

基于 L 系统的方法是目前最为常用的植物建模方法，它可以用来建立植物较为完整的几何模型. L 系统是由生物学家 Aristid Lindenmayer 在 1968 年提出的，最初的应用是对简单的多细胞生物体的发展过程进行建模，随后，L 系统的应用范围逐渐向具有复杂枝杈结构的高等植物扩展[1].

由于使用 L 系统对植物进行建模时，植物模型的几何形状由 L 系统生成的字符串决定，因此要建立不同外形的植物，就要在 L 系统中定义不同的规则. 但是在 L 系统的具体实现中，一般都是针对单独的某一种规则来完成规则的替换，这样，当需要对植物外形进行调整或建立另一种外形的植物模型时，就必须重新实现 L 系统. 针对这一问题，提出一种较为通用的基于 L 系统规则语言分析器的解决方法.

1　基于 L 系统的植物几何建模的基本原理

使用一些等式来描述物体的尺寸、位置和形状，这种表示物体和对物体建模的方法属于欧氏几何方法. 欧氏几何方法比较适于描述表面光滑、形状规则的物体，而对于一些具有整体和局部之间自相似特点的不规则物体，一般使用过程对其进行描述[2]，该过程对产生物体局部细节指定一重复操作，即，若 $P_0=(x_0,y_0,z_0)$ 是选定的初始点，每次重复变换函数，生成的后继层为 $P_n=F(P_{n-1})$. 每次重复时，可以用固定的或随机的生成过程，变换函数可以定义为几何变换（对称、旋转、平移），或者用非线性变换和决策参数来建立.

上述变换函数是一个广义的函数，在将过程方法应用于图形的生成时，经常使用一组产生式规则来作为该函数的定义. L 系统就是定义了一组特定的产生式规则和表示符号，并将其应用到初始物体来增加与原形状一致的层次细节，从而达到描述物体形状的目的. L 系统的核心是重写概念[3]，其基本思想是使用一系列重写规则或产生式规则连续对简单物体的部件进行替代以定义复杂的物体. 该重写过程是一个递归过程. L 系统的重写过程与其他重写过程相比，最显著的特点是规则使用的并行性，即可以同时替代一个给定语句的所有字符. 最简单的 L 系统是确定性上、下文无关的 L 系统，简称为 D0L 系统（deterministic and context-free L system）. 确定性是指每一个字母表中的字符有且只有一条重写规则，上下文无关性是指每一条重写规则只考虑单个字符，不考虑与该字符相邻的其他字符的情况.

L 系统在计算机图形生成，特别是在产生分形和真实感植物建模方面有广泛的应用. 很多分形物体可以看作是一系列基础几何图元的排列，只要给由 L 系统产生的字符串赋予必要的几何信息，即定义一定的形状语法，就可以使用该系统建立分形物体，通常使用导龟几何[1]来解释 L 系统. 对于植物建模[1]，一般用“模块”这一术语来表示在植物生长过程中不断重复的构造单元，例如花苞、花朵、树枝等. 模块级的建模是指对植物生长进行整体描述，即通过描述单个构造单元的生长整合，组成一个特定植物的形状，其过程可用一个使用子模块代替父模块的并行重写系统来表示；所有模块属于一个有限的模块类型的字母表，使用一系列重写规则来生成任意大的各种模块的配置行为. 下面给出一个具体的例子来说明如何使用 L 系统建立植物模型.

L 系统的定义为

$\tilde{\omega}$:F　p1:F→F[+F][−F]

其中，$\tilde{\omega}$ 为开始状态，p1 为规则.

形状语法的定义为：

F：代表植物的单个枝，用圆柱体表示，导龟向前走一个单位的距离.

+：导龟的头部向左偏转一个固定角度.

−：导龟的头部向右偏转一个固定角度.

[：将当前导龟所处的状态压入堆栈.

]：从堆栈中取出状态，并置为导龟的当前状态.

根据上面的 L 系统，可得到如下的推导过程：

F→F[+F][−F]→

F[+F][−F][+F[+F][−F]][−F[+F][−F]]→…

再由形状语法的定义，得到如图 1 所示的结果.

图 1 使用 L 系统生成的植物示意图

2 基于 L 系统的植物几何建模通用方法

针对目前基于 L 系统的植物几何建模方法缺少重用性的问题,提出了一种较为通用的方法,该方法首先对用来进行规则定义的语言进行归纳和抽象,得到可以定义多种 L 系统规则的语言 L-plants,给出由它的标识符、无符号数、保留字、字符分界符等单词符号的定义组成的词法规则,由它的各种语法成分的定义构成的语法规则以及与这些规则相联系的动作,借助 LEX 和 YACC 生成该规则语言的分析器;然后,读入待分析的规则,用上步生成的分析器对其进行分析、解释,完成指定次数的规则替换,得到结果字符串;最后,从左向右,用形状语法对该字符串进行解释.

L-plants 分析器的构造是通用建模方法的关键,也是难点所在.

2.1 LEX 和 YACC 简介

LEX 是个词法分析程序的生成程序[4],它接受一个关于某种语言的面向问题的高级说明,用于字符匹配检查,并提供相关处理动作的源,自动产生可对该语言编写的文件进行词法分析的分析器. LEX 的源由正则表达式和与它相对应的程序片断所构成,源经过 LEX 被转化为一个程序,它负责读输入流,将其复制到输出流中,并且把输入流划分成若干个同已给出的正则表达式相匹配的字符串. 每当识别出一个这样的字符串,其就执行与该正则表达式对应的程序片断. 正则表达式的识别工作是由 LEX 生成的有穷确定性自动机来完成的. 用户写的程序片断按照输入流上对应正则表达式出现的次序执行.

YACC 是 Johson 在 1975 年开发的一个语法分析器生成器[5],它接受一个文法以及和文法相联系的动作,生成能分析该文法所描述语言的语法分析器的分析表. YACC 生成的分析器可连同词法分析器一起使用. 语法分析器调用词法分析程序读输入的源程序,并将单词符号返回给语法分析器. 对于文法的每一条规则,用户可以给出一节代码以指定语义分析和代码生成的动作.

由于 LEX 只接受正则表达式,而 YACC 写出的语法分析程序可以接受很大一类上下文无关的语法,同时它又需要一个低级的分析程序为它识别输入中的单词,因此将 LEX 和 YACC 结合使用,结构如图 2.

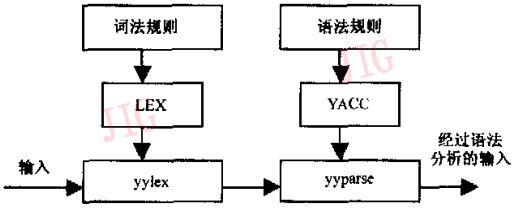


图 2 LEX 和 YACC 的结合

2.2 L-plants 分析器的构造方法

2.2.1 L-plants 分析器的系统设计

L-plants 分析器由词法分析器和语法分析器两部分构成. 词法分析器将扫描 L 系统的规则文件,按照语言的词法规则将有关的字符组合成单词符号,进行词法检查,并将识别出的各类单词符号返回给调用它的语法分析器. 语法分析器根据词法分析器返回的单词符号建立符号表,并将通过计算得到的变量的值也填入符号表的相应位置,识别出 L 系统的开始状态,以及各个规则的前导、条件和后果,并进行存储. 整个 L-plants 分析器向外提供了一个简单的接口,方便应用程序对其进行调用,完成使用识别出的规则进行规则的替换工作.

2.2.2 L-plants 词法分析器和语法分析器的构造

(1) L-plants 词法分析器的构造

在 L-plants 词法分析器构造过程中,由于使用

了 LEX，因此最主要的工作是对 L plants 的词法规则进行定义.

首先，从多种 L 系统的规则，包括最简单的确定性上下文无关的规则、带有参数的规则、带有条件判断的规则、上下文相关的规则和带有随机参数的规则出发，抽象出一套可以覆盖上述所有规则定义的规则语言 L-plants；然后，对该语言的单词符号作如下方式的划分：标识符、无符号数、保留字和字符分界符，定义用正则表达式表示的词法规则，并根据单词符号的种类返回其代码和值[6].

在 L-plants 的词法规则定义中，可能会遇到与左文相关性的问题，即在不同的当前状态下，对同一词法规则或一些有二义性的词法规则，必须采取不同动作的问题. 使用在规则前加"开始条件"的方法来解决这个问题[6]. 在使用开始条件时，必须在定义段中把每个开始条件引入 LEX，在规则前可用尖括号〈〉引用条件，即

〈条件 1〉表达式

是一条规则，仅当 LEX 在该开始条件下才识别它.

下面按照 LEX 源文件的书写规则给出一个在构造词法规则中使用的片断来对该方法进行说明：

```
数字[0~9]
%START  函数  表达式
%%
〈函数〉{数字}  {Save(yytext);return 标识符;}
〈表达式〉{数字}  {Save(yytext);return 整数;}
%%
```

上面%START 后面的"函数"和"表达式"为 2 个开始条件，当 LEX 进入开始条件"函数"时，对单个数字进行识别，并把它作为函数名返回；当 LEX 进入开始条件"表达式"时，对一到多个数字组成的符号串进行识别，把它作为整数返回.

在本文的 L-plants 分析器中，各种不同状态之间的转换由语法分析器控制完成.

(2) L-plants 语法分析器的构造

L-plants 语法分析器的主要功能是识别出 L 系统规则文件中的开始状态和规则，为下一步根据这些信息完成从开始状态向结果转换过程中的规则替换作好准备.

同词法分析器的构造类似，在语法分析器的构造中，由于使用了 YACC，因此只需要定义语法规则和与之相联系的语义动作.

首先，定义 YACC 接受的 L-plants 的语法规范说明中的语法规则. 可以将 L-plants 定义的 L 系统的规则文件看做是由多行内容构成的普通文本文件，把行作为其构成的基本单位，可得到如下用巴科斯范式表示的语法规则：

```
〈L system〉::=〈text_lines〉
〈text_lines〉::=〈text_lines〉〈one line〉|ε
〈one line〉::=〈define_const〉|〈Axiom〉|〈rule〉|'\n'
〈define_const〉::=DEFINE 〈var name〉〈expression〉
〈Axiom〉::=W0':'〈functions〉
〈rule〉::=〈rule_number〉':'〈predecessor〉 ARR()W
〈successors〉|……
……
```

其中，带有尖括号的单词为非终结符，大写字母单词表示终结符. 因规则语言来自对多种 L 系统规则的抽象，所以规则的表现形式多种多样，例如：

(1)上下文无关的确定性 L 系统规则

p1:F→F[+F][−F][F]

(2)带参数的 L 系统规则

p2:F(a)→F(0.5×a)

(3)加入条件判断的 L 系统规则

p3:A(d):d>0→F(1)[+A(d−1)][−A(d−1)]

(4)带有随机参数的 L 系统规则

```
p4:F →(0.6)F[+F]F[−F]F
     →(0.2) F[+F]
     →(0.2) F[−F]
```

(5)上下文相关的 L 系统规则

```
p5:F〈0〉0→0F
……
```

因此，在定义〈rule〉时，对可能产生的所有 L 系统规则形式都要考虑到. 一般地，对仅包含与上下文无关的 L 系统规则的〈rule〉都比较简单，但如果需要考虑上下文相关的 L 系统规则的定义，就需要注意解决二义性的问题. 例如希望在〈predecessor〉中定义包含上下文相关的规则信息，最直观的想法是：

```
〈predecessor〉::=〈functions〉〈context〉
〈context〉::='<'〈function〉|'>'〈functions〉|
            '<'〈function〉'>'〈functions〉|ε
```

但在使用〈context〉::='<'〈function〉和

〈context〉::='<'〈function〉'>'〈functions〉这两条规则时，会出现冲突，要消除冲突，可将规则作如下修改：

```
〈predecessor〉::=〈functions〉〈context〉
〈context〉::='<'〈function〉〈tmp〉|〈tmp〉
〈tmp〉::='>'〈functions〉|ε
```

在完成 L-plants 的语法规则定义后，需要定义

与语法规则相联系的动作,将已识别出的开始状态和规则存放起来,供应用程序使用.另外,建立符号表也是在这些动作中完成的.

2.3 L 系统规则替换

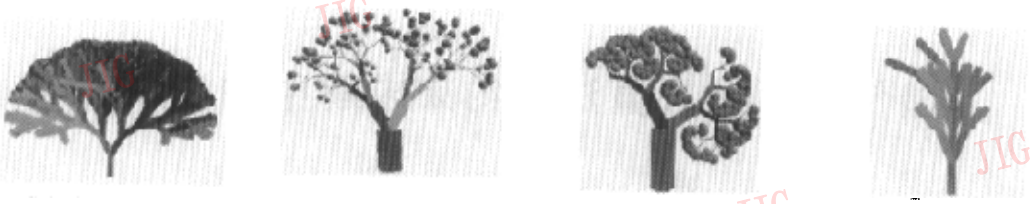
L 系统规则替换过程虽然是在应用程序中完成的,但它却与 L-plants 分析器分析的结果密切相关.一般地,将规则替换次数 n 也按照常量定义的方式写在规则文件中,因此在进行规则替换时,首先需要在由语法分析器产生的符号表中,按照名字查找该数字 n;然后,从左到右扫描开始状态,与已得到的规则列表中的规则逐个进行比较,看是否匹配,如果找到匹配的规则,要进行规则的替换,如果找不到对应的规则,就保留原符号;重复 n 次即得到最后的结果.在进行匹配和进行规则替换时,由于要使用一些变量的值进行条件判断和计算,因此需要使用符号表.

3 实验结果

使用本文方法,对多种 L 系统的规则进行了实验,这些规则的种类包括确定性上下文无关的规则、带有参数的规则、带有条件判断的规则、带有随机参数的规则和上下文相关的规则,下面给出各个规则(表 1)和与其对应的三维几何模型结果[1](图 3).由于本文介绍的重点内容在于 L 系统规则语言 L-plants分析器的构造,所以在进行实验时,使用的形状语法都比较简单,因而获得的植物模型不是特别精细.如果要得到更为精细的植物模型,只需对形状语法进行一些必要的修改即可.

表 1 L 系统规则

规则 1	Axiom:F p1:F→F[+F][///+F][\\\F][-F]
规则 2	#define r1 0.8 #define r2 0.8 #define a1 30 #define a2 −30 #define b1 137 #define b2 137 #define w0 30 #define q 0.5 #define min 20 Axiom: A(100,w0) p1:A(s,w):s>=min→! (w)F(s)[+(a1)/(b1) A(s*r1,w*q)][+(a2)/(b2)A(s*r2,w*(1−q))]
规则 3	#define r1 0.65 #define r2 0.68 #define a1 27 #define a2 −60 #define b1 0 #define b2 10 #define w0 20 #define q 0.53 #define min 1.7 Axiom:A(100,w0) p1:A(s,w) →(0.85) ! (w)F(s)[+(a1)/(b1)A(s*r1,w*q)] [+(a2)/(b2)A(s*r2,w*(1-q))] →(0.07) ! (w)F(s)[+(a2)/(b2) A(s*r2,w*(1-q))] →(0.13) ! (w)F(s)[+(a1)/(b1)A(s*r1,w*q)]
规则 4	#ignore +−F Axiom:F 1 F 1 F 1 p1:0<0>0→0 p2:0<0>1→1[1+/F 1 1 F 1] p3:0<1>0→1 p4:0<1>1→11[1+F 0 F 1][-//F+ +F] p5:1<0>0→0 F 0 p6:1<0>1→1 F 1 p7:1<1>0→1[0+F 1 F 0] p8:1<1>1→0 p9:+→− p10:− →+



(a) 据规则 1 建立的植物模型　(b) 据规则 2 建立的植物模型　(c) 据规则 3 建立的植物模型　(d) 据规则 4 建立的植物模型

图 3　根据各规则建立的模型

4　结　论

针对使用 L 系统进行植物几何建模的具体过程随规则定义的变化而变化的问题，提出了一种较为通用的基于 L 系统规则语言分析器的解决方法：通过归纳和抽象得到可以定义多种 L 系统规则的语言 L-plants，并为其构造语言分析器，完成 L 系统开始状态和规则的识别，以达到规则替换的目的，最后使用形状语法对字符串进行解释，建立几何模型. 经过实验，该方法可以对 L 系统的多种规则进行正确的识别，让人们在建立各种造型的植物模型时可以将注意力更多地放在 L 系统规则的定义上，而不需要考虑规则替换过程的细节，确实能够提高植物几何建模的效率. 该方法的另一个特点是具有良好的可扩展性，如果需要识别其他类型的 L 系统规则，只需要在语法规范说明文件中添加适当的语法规则. 另外，本文方法的重点在 L 系统规则语言分析器的构造上，而植物建模只是 L 系统的重要应用之一，所以本文方法是否可以用于基于 L 系统的其他应用还有待进一步的研究.

参 考 文 献

1　Prusinkiewicz P, Hammel M, Hanan J *et al*. L-systems: From the theory to visual models of plants [EB/OL]. http://www.cpsc.ucalagry.ca/Research/bmv/papers/index.html, 2001-10-10/2002-5.

2　Donald Hearn, M. Pauline Baker. Computer Graphics [M]. New Jersey: Prentice-Hall International Inc, 1997.

3　Gabriela Ochoa. An introduction to lindenmayer systems [EB/OL]. www.cogs.susx.ac.uk/lab/nlp/gazdar/teach/atc/1998/web/ochoa/, 1998-12-02.

4　Johnson S. YACC—Yet another compiler compiler [R]. Computing Science Technical Report 32. Murray Hill, USA: AT&T Bell Laboratories, 1975.

5　Lesk M, Schmidt E. Lex—A lexical analyzer generator [R]. Computing Science Technical Report 39. Murray Hill, USA: AT&T Bell Laboratories, 1975.

6　高仲仪，金茂忠. 编译原理及编译程序构造 [M]. 北京：北京航空航天大学出版社，1996.

王莉莉　1977 年生，1999 年获北京航空航天大学计算机科学与工程系学士学位，1999 年开始攻读京航空航天大学计算机科学与工程系硕士研究生，2000 年转为博士研究生. 主要研究方向为虚拟现实.

赵沁平　1948 年生，教授，博士生导师，1986 年获南京大学计算机系计算机理论专业工学博士学位. 主要研究方向为虚拟现实、可视化和人工智能等.