3D models generator simulating a grow of natural objects for virtual reality

Lukáš Szemla
xszeml00@stud.fee.vutbr.cz

Department of Computer Science and Engineering
Brno University of Technology
Brno / Czech Republic

Abstract

The goal of this work is to design and prove basic principles of fractal generating of natural plant objects like trees, bushes, etc. This work discusses one possible way of creating models of plants based on L–system gramatics and deformation functions development. Whole work is intended to be a support for virtual reality systems.

KEYWORDS: natural object, plant, skeleton, L–system, grammar, SDL, deformation functions.

1 Introduction

Grow simulation generator of 3D models of natural objects is tool for creating models of natural plants. This document presents one possible solution of this problematic.

If you want to simulate some real scenery, e.g. landscape, you will probably want to use realistic looking models of plants. To create a good working generator of those plants, which will produce models of good quality, is not easy task when you think about variety of natural plant objects and it pays to devote care to design strong principles of creating plant models. That is a reason why I have consecrated my work to this problem.

Methods for modeling and generating models of plants should be effective, easy to use and, at once, strong enough to ensure satisfactory quality of produced models which will not be very hard to visualize. Full 3D models are required for needs of virtual reality and not only set of 2D structures simulating 3D model. It is is unacceptable because those set of 2D structures has not important features which are necessary for good 3D visualisation (shadows, prespective, etc.)

Process of plant design and generating is divided into two phases. So called *skeleton* of natural plant object is designed in phase one. Problem of skeleton design is discussed in section 2. The skeleton is developed into final representation of model in phase two. Methods of skeleton development are described in section 3.

2 Skeleton design

2.1 What the skeleton is

Skeleton of natural plant object is an abstract object which determines space lay–out of future model (that means its dimensional proportions). Skeleton also tells about dependences of individual parts of plant model. Skeleton consist of parts called *compponents* which are its elementary indivisible units. Skeleton determines character and fundamental look of future model and that is why it's so important. One must consider carefully skeleton design to obtain model of good quality.

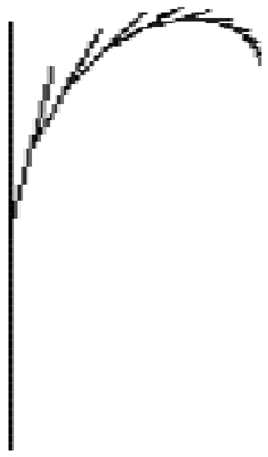

Figure 1: Skeleton of simple straw.

2.2 Methods of skeleton design

There are more ways how to design skeleton of plant model. I have created desription language based on *Lindenmayer systems* (L–systems). This language is called *SDL* (Skeleton Description Language) and this paragraph is discussing principes of skeleton design using SDL. L–systems come from formal grammars and they offer simple and strong enough mechanism for modeling living orgamisms in general. It is necessary to bring this formalism into use. This document is not intended to exactly explain how does L–systems work. For further information about L–systems see [1].

As mentioned in section 2.1 skeleton consists of components and determines their dependences. Components are elementary indivisible units but they do not need to have the very same features. That is why it is useful to divide skleleton design into two parts. Features of components are specified in th first part and their dependencies are described in th second part. I will talk about second part at first.

2.2.1 Component dependencies description

Description component dependencies is part of skeleton design which is based on L–systems. Grammar and its rules are used for determination of dependencies between components. Rules of this grammar are consisted of nonterminal symbols only. Each nonterminal symbol represents one component and rules describe their dependencies. In other words, rules mean *patterns* which will be used for skeleton construction. Deriving mechanism of this rules is the same as deriving mechanism of formal grammars. That means that during proces of derivation the nontermial symbol in output sequence is replaced by its right side of relevant rule.

Meaning of nonterminal symbols is rather different from meaning of nonterminal symbols in formal grammars. Each nonterminal symbol represents one component described by its features. Features of components will be discussed in section 2.2.2. It is necessary to understand features passing at this time. Relations of components are described by rules of grammar of L–system. *Master* component stays on left side of this rule and *slave* component (or components) stays on right side of relevant rule. That means that during process of derivation slave components will deduce their features from master component. For better understanding follow this example. Let's have a rules:

$$S \rightarrow A$$

$$A \rightarrow A$$

Derivation will proceed like this:

$$S \rightarrow A_0 \rightarrow A_1 \rightarrow A_2 \dots$$

Components A_0, A_1, A_2, etc. are derived from the same nonterminal symbol but they have another features. Nonterminal S (representing one compomnent) is starting nonterminal symbol, as you can see, and its rule is used at first. Component A_0 will derive its features from component S in this step. Then the second rule is used repeatedly so that component A_1 will derive its features from component A_0, component A_2 will derive its features from A_1, etc. Note that this process of deriving is neverending and it is not important at this level of skleleton design to stop it. This is a way how to simulate grow of plant because you can stop process of deriving after different count of iterations.

It is usefull to change implicit master – slave relations of nonterminal symbols sometimes. SDL enables this by using so called *technique of branching*. Branching is technique for explicit change of nonterminal dependencies. That means that master nonterminal for some slave nonterminal symbol on right side can be also found on right side. It must be used two *stack symbols* to change master nonterminal for one or more slave nonterminal symbols. This symbols are brackets ("(" and ")"). I will demonstrate this facts on example. Let's have a rule:

$$S \rightarrow (A\ B(CD)E)$$

I use branching in this rule. Nonterminal S is master for nonterminal symbols A, B and E. Symbol B is master for nonterminals C and D. Components repesented

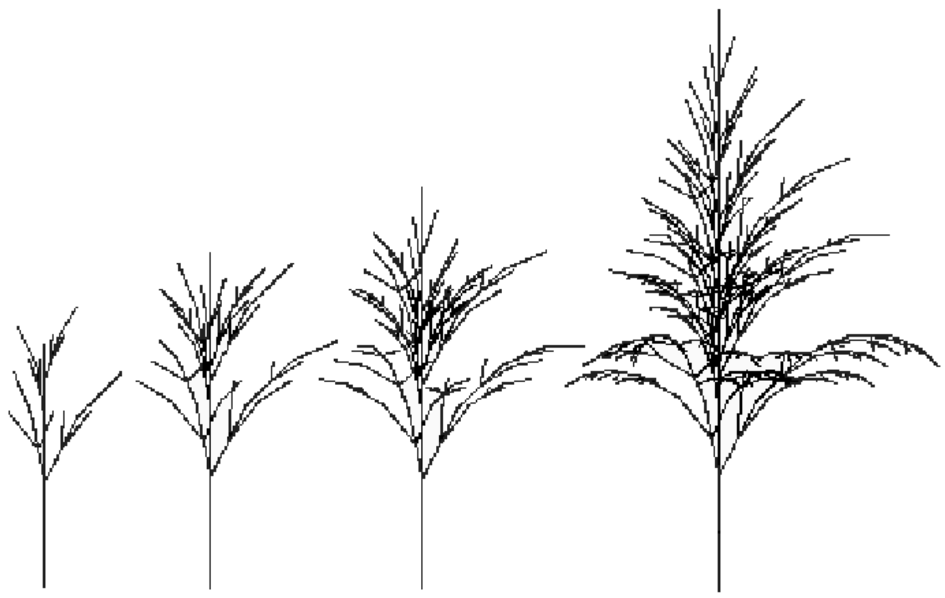

Figure 2: Example of grow of skeleton.

by nonterminals A and B will be created during process of derivation at first, and component B will be used as master for components C and D immediately after. Technique of braching is good for creating more complicated branch structures in one step of derivation process. For further informations see simple example in section 2.2.4.

2.2.2 Component features description

As mentioned before, components represented by nonterminal symbols have some features. You can read some facts about component features in this section. Features are very important for model design and they determine character and look of fututre model. Some basic atributes are needed to describe curves (for examle: skeleton components are represented by curves) in 3D scene. SDL supports four features at this time. There is begining of component, lenght, vertical and horizontal angle of rotation. It is very important to know that feature of component is specified as relative difference from master component feature. In other words, all atributes of component feature must be considered as a value of variation from relevant atribute of master component. This is one of the most important principle of SDL. Now I can describe meaning of each component in this context.

Atribute of begining means location where slave component begins on master component. Atribute of lenght determines lenght increment or decrement of slave component with respect to master component and angles of rotation are determined in the same way. Absolute values of component features are determined during process of derivation and we can talk about it as about the process of interpretation in this context. Determination of absolute values of component features is the main difference between derivation and interpretation, if we talk about derivation we mean replacing nonterminals only.

SDL also offers methods to achieve variability of constructed models at this level of skeleton description. This method is called *mutating*. You can delimitate valid interval for some atribut values instead of one value only. One value from this interval will be choosen accidentally in each step of process of interpretation. One or more atributes of component feature can be signed as mutating, e.g only atribute of horizontal rotation angle can be assigned for mutating. (see Figure 3)

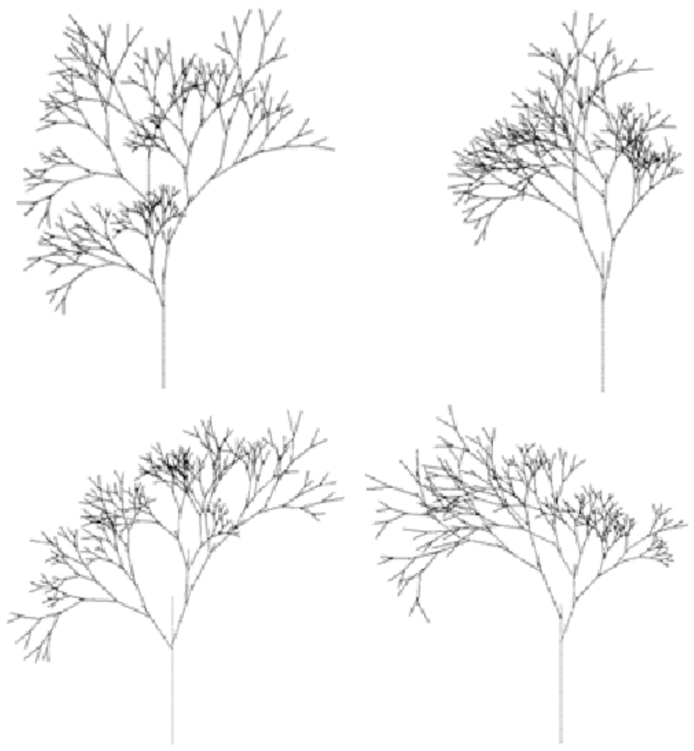

Figure 3: Example of mututating.

2.2.3 Syntax of SDL

Exact specification of SLD syntax using grammar:

$$\begin{array}{rcl}
S & \rightarrow & define\ \{A \\
A & \rightarrow & LB;\ A \mid \}F \\
B & \rightarrow & id\ [\ C \\
C & \rightarrow & XD \mid NULL] \\
D & \rightarrow & ,E \mid] \\
E & \rightarrow & XD \\
F & \rightarrow & rules\ \{G \\
G & \rightarrow & H;\ G \mid \} \\
H & \rightarrow & id -> \ I \\
I & \rightarrow & (J) \\
J & \rightarrow & id\ KJ \mid e \\
K & \rightarrow & I \mid e \\
L & \rightarrow & \#const\_ver \mid \#const\_hor \mid e \\
X & \rightarrow & numY \\
Y & \rightarrow & :Z \mid e \\
Z & \rightarrow & num
\end{array}$$

I can write a few words about SDL syntax at this time. I do not want to be much concrete because this work is not finished yet and some details can be changed. At this time, SDL source is divided into two parts as mentioned in section 2.2. These parts are called *blocks*. First block is introduced by keyword `define`. This block is obligatory and you must describe component features in this block. First, the name of nonterminal symbol is specified and then description of feature. Feature is bounded by symbols `[` and `]`. You need to specify begining of component, length, vertical and horizontal rotation angle in this order. You can specify feature either as one value or as interval for mutating. This interval is described by middle value and maximum deviation from middle value separated by symbol "`:`". Second block, that follows, is block introduced by keyword `rules`. This block is also obligatory and you can describe component dependencies in this block. This is done by writing rules. You specify left side of rule at first and then right side of relevant rule. Only one nonterminal symbol can stay on left side of rule. Left and right side are separated by symbol `->`. You must use *stack symbols* and then to specify master component even if you do not want to change implicit master – slave component relation (see section 2.2.1). You can see some symbol in SDL grammar as `#const_hor`, `#const_ver` or `NULL` symbol. Meaning of this symbol is topic of evolution and future of this symbols is unwarranted.

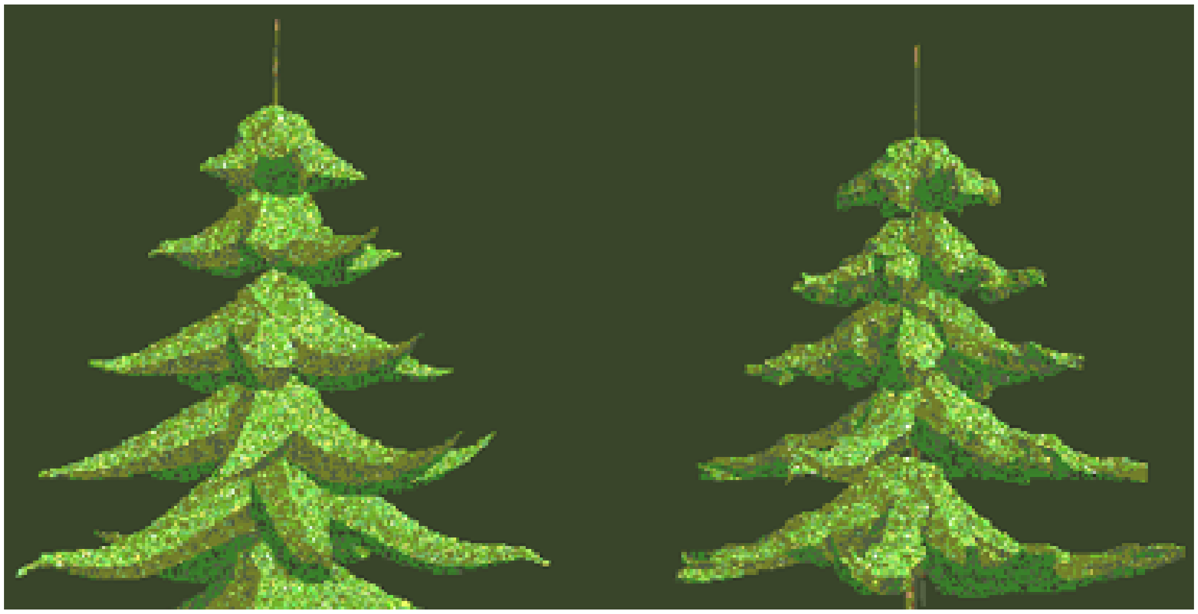

Figure 4: Example of use of correction function.

2.2.4 Simple example

I will demonstrate all described facts on simple example. Figure 1 show result of interpretation of following simple SDL source code in few generations.

```
// source for describing simple straw skeleton
 define{
   start[0,1,0,0];
   straw[0.5:0.3,0.8:0.3,0,30:10];
}
rules{
   start -> (straw);
   straw -> (straw);
}
```

3 Skeleton development

So far we know how to describe and create skeleton of natural plant object using SDL. It is very important part of model design but it is not all what should be done. As mentioned in section 2.1, skeleton of plant is an abstract object which can be presented by curves. But real plants are not curves as we know. So it is necessary to transform skeleton into more realistic looking model. This transormation is called *skeleton development* or *skeleton expansion*. This section describes methods and principles of skeleton expansion.

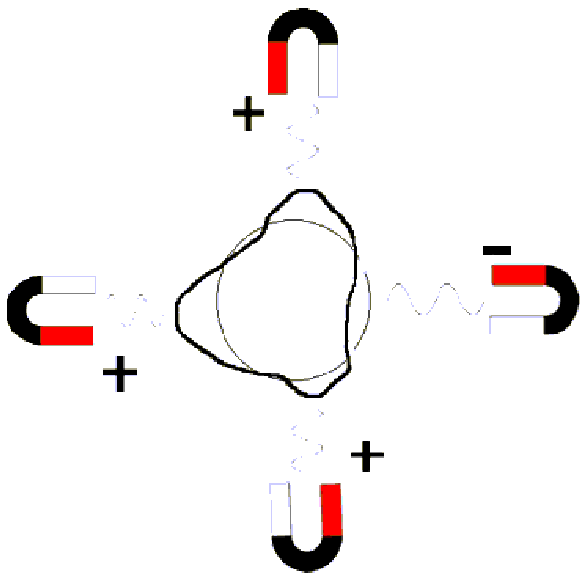

Figure 5: Principle of magnetic function.

Basic idea in skeleton expansion is to transform curves to some 3D object and then deform this object. The usable 3D object can be a cylinder with circular or elliptic cut and it is called *fibre* in this context. Fibres are deformed by *deformation functions*. There are three categories of deformation functions and each category is defined to change another feature of fibre. This categories are contour functions, magnetic functions and correction functions. Contour functions are aimed to bend fibres without change of its cut. Magnetic functions are identified to change cut of fibre. Idea of magnetic function is as follows. Fibre is made from limitless elastic and magnetic material and its proportions can be changed by magnets. Several magnets with different magnetism are placed around fibre's cut (see Figure 5) and this magnets change cut of fibre. Those rings of magnets are placed along whole fibre and they change cut of fibre in all locations. Magnetic function is designed to change basic shape of fibre. Correction functions are aimed to make deformed fibre looking better (see figure 4). Correction functions cooperate with magnetic functions and they roughen surface of deformed fibre. They are not intended to be used for expressive change of fibre shape but to change its surface. Proper use of deformation functions is highly recommended.

Technique of design of deformation functions enables us to have either flat or solid fibres (see Figure 6 and Figure 7). It is left on author's consideration, if he whish to have flat fibres or not. So called *materials* are also created and mapped during process of skeleton development.

Material determines final surface adjustment and it contains texture mapping, light emission, transparency, etc. In addition, two types of fibres are distinguished in plant model. They are

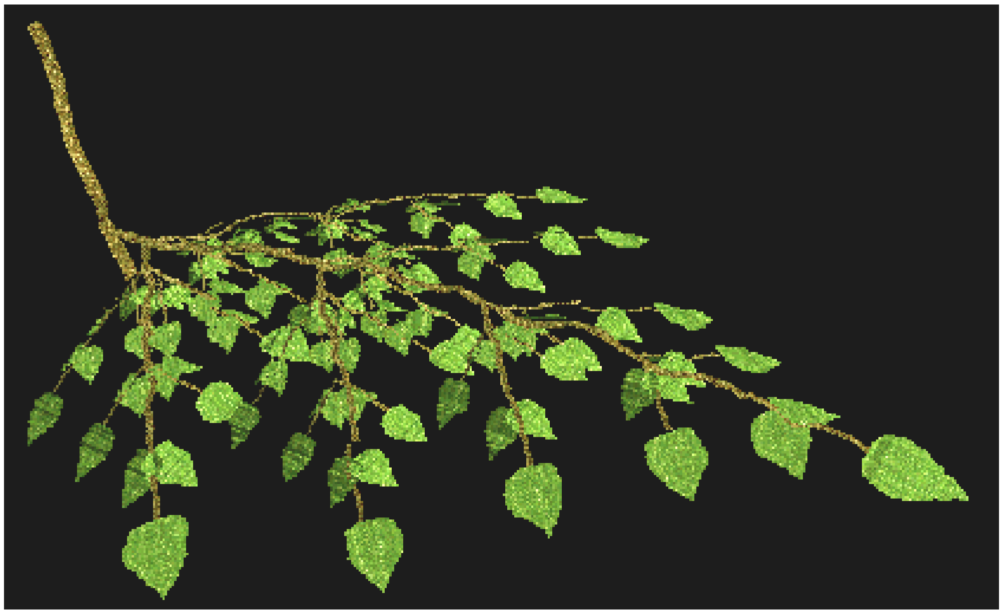

Figure 6: Example flat leaf fibres.

so called *wooden fibres* and *leaf fibres*. In other words, you can use another sets of deformation functions, materials or prescribe flatness or roundness for each type of fibre separately. That enables us to design different looking parts of plant that have another features.

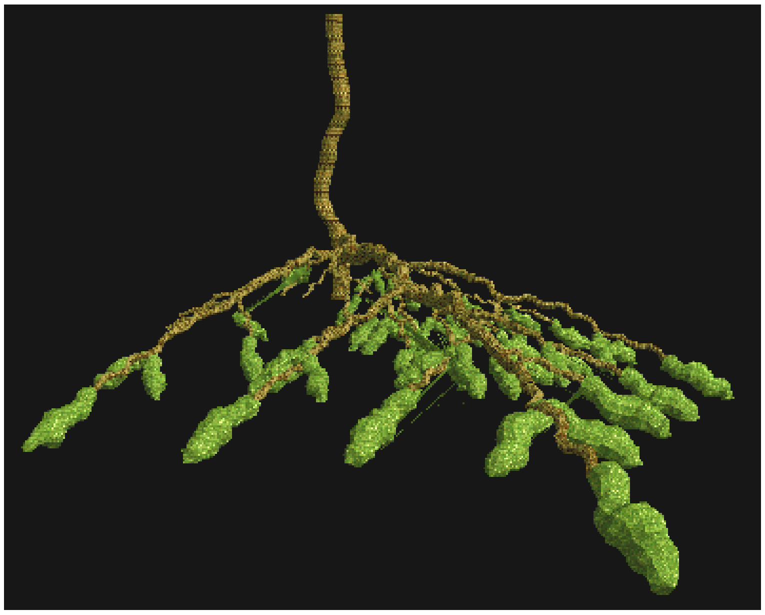

Figure 7: Example solid leaf fibres.

Design of deformation functions has also influence on quality of future plant model. It is a proportion between number of used magnetic funtion and quality of model. More magnetic functions in ring means better quality but those models also consume more memmory. Consumption of memmory is still big problem because models are either to big to be visualised so fast as we wish or they are not looking so good.

4 Conclusion

This work is enganging in natural plants modeling and shows some of its bacis principles. It is more ways how to construct models of natural plants. One of them is using formalism of L–systems and I decided for this way which appears to be usefull and perpective. My work is not completly finished and I have to consider some principles of SDL language and I want to add new features like cycles, possibility to define atributs of components as a functions or enable more than one right side. I want also to improve some features of skeleton expansion and deformation functions.

References

[1] Prusinkiewicz P. Lindenmayer A.: *The Algorithmic Beauty of Plants*, Springer–Verlag, New York, 1990

[2] Prusinkiewicz P., Hanan J.: *Lindenmayer systems, fractals, and plants*, Springer–Verlag, Berlin, 1989

[3] Mandelbrot B.: *The Fractal Geometry of Nature.*, W.H. Freeman, New York, 1975

[4] Szemla L.: *Generation of natural plant objects*, Brno university of technology, Brno, 1999