Procedural fractal plants generation

Filip Rynkiewicz
Institute of Information Technology, Lodz University of Technology, ul. Wolczanska 215, 90-924 Lodz, Poland
173186@edu.p.lodz.pl

Piotr Napieralski
Institute of Information Technology, Lodz University of Technology, ul. Wolczanska 215, 90-924 Lodz, Poland

Abstract

Since the beginning of computer era we are trying to imitate nature. The results of those attempts are mathematical equations describing weather, snow flakes, plant growth or influence of species in individual biomes. In computer graphics fractal structure are often used because they can be simple characterize in mathematics. Those forms are common in nature. Self-similar fractals are created in computer graphics using Lindenmayer Systems. This article was formed to analyse efficiency of this algorithm in modelling 3D tree triangle meshes for games. The general motivation of this topic is to produce way to procedural modelling and simplified edition of complicated 3D tree models. By combining L-systems with Bézier curve there is a possibility to produce complicated 3D models. For example tree created with 468 branches having 87200 vertices and it is made in few seconds. In conclusion, time-consuming modelling in 3D programs can be replaced by solution described in this article.

1 Introduction

Using 3D modelling programs to create a complex 3D mesh of tree is very time consuming, because every branch has to be shaped separately. To speed up and automatize this process Lindenmayer System can be used. Since 1968, when Aristid Lindenmayer created grammatic to represent simple multicellular organism, technique has evolved. Now L-systems are used to manage 3D models of plants[1, 3, 5, 14, 4], fire visualization[2], music[6] or even neural web[7]. Variaty os usage, flexibility and simplicity of this technique creates a possibility to extend it. Developers over years have used this technique to create software like L-studio[8], plugins to 3DsMax to animate plant growth[9] or in Houdini[11].

Most of the algorithms to create 3D tree mesh are generating static models, to change or move branch user must change parameters inside L-system, and recalculate whole model. To prevent unnecessary actions and calculation the *Bézier curve* was added as the tree skeleton. Adding possibility of changing Bézier points along Bézier curve, user can change every branch only by moving it's handler and recalculating only this alternated branch.

After long years of plants observations the conclusion was brought, every plant can be treated as a very complex fractal. The most important feature of this structure is self-similarity. In mathematics, a self-similar object is exactly or approximately similar to a part of itself (i.e. the whole has the same shape as one or more of the parts)[10]. Because of complexity of plants as fractals the simplifications must be applied. To do this the L-system can be used. Mentioned above technique is based on rewriting rules, so on replacing sub-terms of a formula with other terms. All formulas and terms are represented as strings, and L-system is all about changing one string to another based on rules. Every system has to have starting word, called axiom, rules of productions and number of iterations, so how many times rules must be applied to word.

2 Procedural systems

Creating data algorithmically, based on mathematical equations, is the core of procedural system. This mean that program creates content on its own, or with limited or indirect user input. In case of L-systems there are few possibilities of creating procedural content using simple or more complicated L-systems.

2.1 OL-system

Most basic L-system is context-free L-system, called OL-system, where there can be multiple replacement rules to any given value. In example:

$$\begin{aligned} A &\longrightarrow \alpha \\ A &\longrightarrow \beta \end{aligned} \tag{1}$$

That means that A can be replaced with either α or β.

When there is one and only one replacement for the same letter in word, the system is called *DOL-system*, deterministic OL-system. Giving *axiom*(starting word) as b, rules : $b \rightarrow a$ and $a \rightarrow ab$ with 5 iterations simple DOL-system can be created. Example production of this system is shown at Figure1.
Because L-systems was inspired by nature, usage of it can be found in there. In example development of cyanobacteria *Anabaena Catenula* can be simulated by DOL-system.

Figure 1. Example of DOL system[12]

2.2 Turtle interpretation of string

L-systems are operating on symbols, so it's mean that it will always creates list of characters. Every symbol in created word can be classified as *steering character*, in example in Table1.

Table 1. Steering characters

sign	**discription**
+	Turn left by angle δ.
-	Turn right by angle δ
&	Pitch down by angle δ
^	Pitch up by angle δ
\	Roll left by angle δ
/	Roll right by angle δ
|	Turn around, by 180 $^\circ$
[	Add branch
]	End branch
[A . . . Z]	Draw line between points

To create graphical content from it those characters must be parsed to specific action on *turtle*. *Turtle graphics* is term where turtle have it's position, angle which describes the direction in which it is facing and any other parameters that user will use, but position and angle are mandatory. When changing turtle behaviour, so parsing specific character, the different graphical content can be applied.

As it can be seen in the last column in Table1 the steering character to draw line between points can be represented by multiple upper cases letters. User can change behaviour of turtle by manipulating those characters. Can be defined which one are representing draw line and which are just to create randomness of given L-system.

2.3 Bracket L-system

OL-systems always create one long continued line. To add branching like behaviour, new characters must be added. Using *[* and *]* the *branches* can be created. First character is responsible for starting the branch and next for closing it. Branch is simply new line in geometric interpretation of string created by L-system Figure2.

Figure 2. Example of Bracket L-system[12]

2.4 Parametric L-system

To create more complicated models the parametric L-system was developed. Thanks to parameters L-system can change the values of variables, in example the steering angle of branch. When in OL-system the sign + was strictly assigned to hard-coded value, in this approach the value can be changed in every iteration using *+(r)* where *r* is some kind of parameter. Every letter or string must have brackets immediately behind it and within them there can be multiple parameters, separated with comma, also mathematical expressions can be applied between parameters. Example:

$$C(x \cdot e, e \backslash 2, a + c, b - d, g) \tag{2}$$

As can be seen in Equation 2 the operands between parameters can be every mathematical operand that can be applied between two floating point numbers.

3 Bézier curves

Combining all mentioned above L-systems user can create model of plan, but without any possibility to change it without recalculating whole system. To add this feature in this work the Bézier curves was added. It will create whole skeleton of tree, and then based on it the mesh will be created.

Bézier curve is parametric curve, that can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bézier curve can be represented mathematically as

$$\sum_{k=0}^{n} P_i B_i^n(t) \tag{3}$$

Where P_i is the set of points and $B_i^n(t)$ represents the Bernstein polynomials which are given by

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i \tag{4}$$

Where **n** is the polynomial degree, **i** is the index, and **t** is the variable.
A B-spline curve is defined as a linear combination of control points P_i and B-spline basis function $N_i k(t)$ given by

$$C(t) = \sum_{i=0}^{n} P_i N_i^k(t) \quad n >= k-1, \quad t \in [tk-1, tn+1] \tag{5}$$

where

- $$P_i = t_0, t_1, \ldots, n, \tag{6}$$

  P_i describes the control points.

- k is the order of the polynomial segments of the B-spline curve. Order k means that the curve is made up of piecewise polynomial segments of degree $k-1$.

- The $Ni,k(t)$ are the "normalized B-spline blending functions". They are described by the order k and by a non-decreasing sequence of real numbers normally called the "knot sequence".

$$t_i : i = 0, \ldots n+K \tag{7}$$

N_i, k functions are described as

$$N_i^1(t) \begin{cases} 1 & if\, t_i \leq t < t_{i+1}\, and\, t_i < t_{i+1} \\ 0 & Otherwise \end{cases} \tag{8}$$

if $k > 1$

$$N_i^k(t) = \frac{t - t_i}{t_{i+k-1}} N_i^{k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1}^{k-1}(t) \tag{9}$$

also

$$t \in [t_{k-1}, t_{n+1}) \tag{10}$$

4 Methods

To implement program, from which screenshot will be presented ,all mentioned above methods was used. Implementation was write in Unity Engine version of C# called Mono. To do this those custom classes was used:

LType	description of parameter, its definition as float and name of object
LFunction	definition of production and collection of parameters used in this production
LObject	object of L-system
PosRot	quaternion and position

Every *LObject* have list of *LFunction* and Dictionary of *LType*, where *Key* is string name and *Value* is float. Class where are all methods that calculates L-system productions is *Interpreter*, where there are 2 functions *CreateTreeString and CreateBezierTree(LObject currentObject)*. First create word from rules and second creates tree skeleton based on this word.

Word is created by replacing one string by another using rules corresponded to LObject. Because in this implementation only Parametric L-system is used, all parameters must be evaluated in every iteration. There are two main cases to do this. When parameters inside bracket have mathematical expression, and when they don't. Second event is simpler to evaluate because algorithm have to find corresponded value to string representation in Dictionary, and than change it for float value. It's performed only at the beginning of algorithm, and with every function. Thanks to this, algorithm does not need to change every parameter in every iteration, it's done only once. In example Equation 11

$$!(vr)F(50)[\&(a)F(50)A]/(d2) \quad \longrightarrow \quad !(2.13)F(50)[\&(0.45)F(50)A]/(11.3) \tag{11}$$

Parameters that have mathematical expression between them have another approach to evaluate them. At first parameters have to be changed to float values then expression must be specified and the last step is to interpret, calculate value as float and then return it as string value.

$$F(l) \rightarrow F(l \cdot lr) \tag{12}$$

In this example at Equation 12 the rule $F(l)$ must be found in list of production of current *LObject*, l and lr parameters will have to be found in Dictionary of parameters, $\cdot$ sign must be interpreted as multiplication sign, and the last step is to return result as string representation of float, so if $l = 6$ and $lr = 0.32$ the result of this rule would be *F(1.92)*.

Second part of algorithm is to generate tree skeleton, based on created before word. String may consist of several thousand characters, so analysing it is the most time consuming part of this algorithm. Every Bézier curve has it's parameters such as *normal, bi normal and tangent vector*, corresponded to *X,Y and Z position*. Every branch is separate Bézier curve, which consist of Bézier points. Those structures can be moved, deleted or added. That operations are based on steering characters in the word. Last step is to generate mesh based on points of skeleton. Every point on Bézier curve is center of one 3D cylindrical mesh. Normal, binormal and tangend are used in this part to calculate properly bending angles. All calculation are based on Quaternions rotations, to ensure appropriate angles, and because that how the Unity is handling the angles.

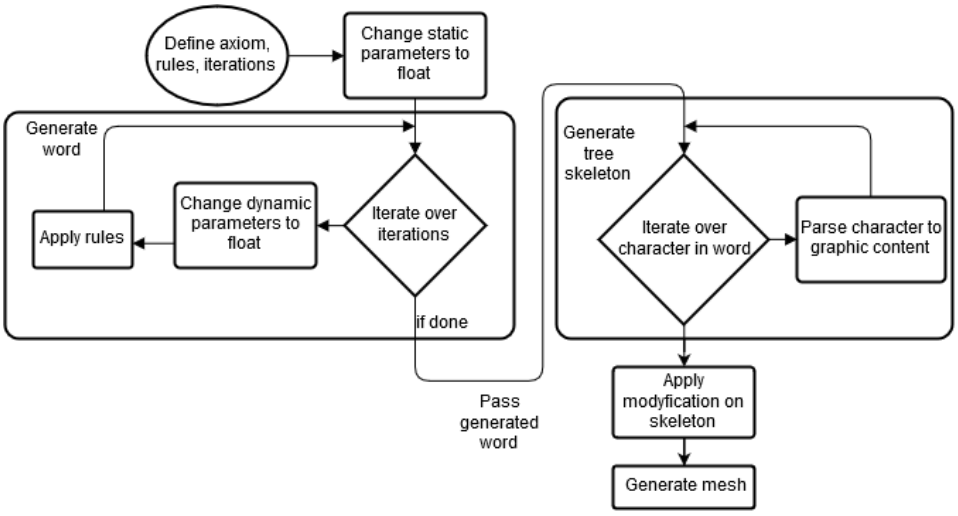


Figure 3. Flow chart of this algorithm

5 Results

For each of the example the same values was used. Five segments on the side surface and five points on spline, it's mean that every curve was divided on five equal parts. Each result of implemented algorithm was divided into two pictures, screenshot of model generated by algorithm and model after changes. Changes was made by transforming Bézier points on corresponded to branch Bézier curve. First all the example and it's description was shown, then the pictures. Description of every example system contains rules, axiom and number of iteration used within it.

Example 1

In first example for 5 iterations the number of branches is 468, word forming tree have 46086 chars and number of vertices in model is 87200.

This example is modification of simple fractal plant described in [12]. Every branch is directed at 25 $^\circ$ relative to parent branch. As it can be seen at the Figure4a, the model created by algorithm is complex, using only two rule and one variable. Model after changes have been shown at Figure4b. Every branch in tree have been altered according to user actions.
Rules:

$$F(l) \rightarrow F(l \cdot 2) \tag{13}$$

$$X(l) \rightarrow F(l)[/(r)X(l)]F(l)[\backslash(r)X(l) - (r)X(l)]F(l)[\backslash(r)X(l) + (r)X(l)] \tag{14}$$

Axiom:

$$\mathrm{X(1)}$$

Number of iterations:

$$4$$

Variables:

$$r \rightarrow 25 \tag{15}$$

(a) Mesh without modification. Own source.

(b) Mesh after modification. Own source.

Figure 4. Comparison of meshes for example 1.

Example 2

Last example have 214 branches, world world forming tree have 24918 chars and 73700 vertices in model. Figure5a presents unchanged model of last example, Figure5b same model after changes.
Rules:

$$F(l) \rightarrow F(l \cdot 2) \tag{16}$$

$$\begin{aligned} X(l,w) \rightarrow\ & F(w \cdot l)[/(r \cdot l)X(l,w) - (r)X(w,w)C(l)] \\ & + (r)F(l)[G(l) \setminus (l)X(l,w) + (r)X(l,w)] - (r)F(w) - (r)F(w) \end{aligned}$$

$$G(l) \rightarrow F(l/5)[X(l,l) + (r \cdot k)] \tag{17}$$

$$C(l) \rightarrow F(l/5)[X(l,l)/(r \cdot k)] \tag{18}$$

Axiom:

X(1,2)

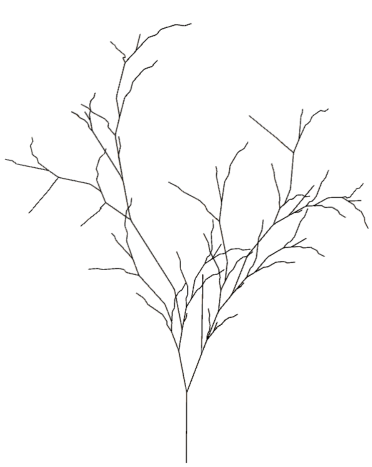
Number of iterations:

5

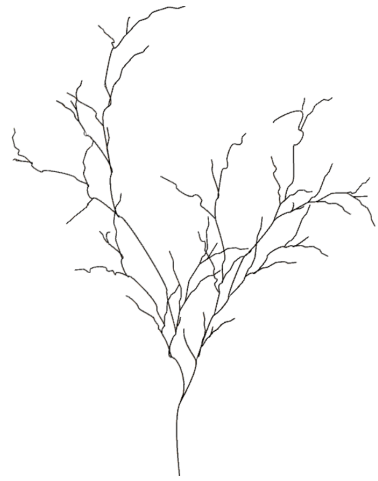
Variables:

$$r \rightarrow 23.5 \tag{19}$$

$$k \rightarrow 0.707 \tag{20}$$



(a) Mesh without modification.
Own source.



(b) Mesh after modification.
Own source.

Figure 5. Comparison of meshes for example 2.

6 Conclusion

By using technique described before the user can simply generate tree-like model in short time, small amount of vertices, and with ability to change every point of model using Bézier handles. Creating model with 300 branches using normal 3d software would be very time consuming.We can move our work to machine and automatize process of modelling, using simple parametrized mathematical equations, without even knowing how the modelling software works. And we can do this inside game engine, without other programs.

To evaluate all string into float values the simple parser have been created. Parser created with this implementation is simple, but creating possibility to evaluate more than 3 parameters, and operations between them, using approaches from this implementation would be very difficult. To ensure that every parameter would be evaluated properly whole implementation will be rewritten into new *Boost Spirit X3* C++ library, and created as *Dynamic-Link Library*. Unity,Unreal Engine and others game engines have ability to write plugins based on *DLL* written in **C++**, so every engine could have their own implementation of this algorithm.

References

[1] Prusinkiewicz P., Lindenmayer A.,Hanan J., *Development models of herbaceous plants for computer imagery purposes*, SIGGRAPH Comput. Graph., New York, 1998

[2] Zaniewski T., Bangay S., *Simulation and visualization of fire using extended lindenmayer systems*, AFRIGRAPH '03, Cape Town, 2003

[3] Qi H.,Qiu R.,Jia J., *L-system based interactive and lightweight web3D tree modeling*, VRCAI '11, Hong Kong, 2011

[4] Zhuming L., King S., *Simulating tree growth based on internal and environmental factors*, GRAPHITE '05, Dunedin, 2005

[5] Hanan J., Renton M., Yorston E., *Simulating and visualising spray deposition on plant canopies*, GRAPHITE '03, Melbourne, 2003

[6] Manousakis S., *Musical L-Systems*, The Royal Conservatory, Hague, 2006

[7] Palmer M., *Evolved neurogenesis and synaptogenesis for robotic control: the L-brain model*, GECCO '11, Dublin, 2011

[8] L-studio, `http://algorithmicbotany.org/virtual_laboratory/`

[9] Bartniak A., *3D animation of plant development using L-systems*, M.Sc. thesis, Lodz, 2013

[10] Mandelbrot, B., *How long is the coast of Britain? Statistical self-similarity and fractional dimension*, Science, vol. 156, no. 3775, 1967, pp. 636–638. http://www.jstor.org/stable/1721427.

[11] Houdini Engine L-System, `https://www.sidefx.com/docs/houdini/nodes/sop/lsystem`

[12] Prusinkiewicz P., Lindermayer A., *The Algorithmic Beauty of Plants*, Springer , 2004

[13] Bézier curve, `http://mathworld.wolfram.com/B-Spline.html`

[14] Fuhrer M. Hairs, *Textures, and Shades: Improving the Realism of Plant Models Generated with L-systems*, M.Sc. thesis, University of Calgary, 2005

[15] Rozenberg G., Salomaa A. *The mathematical theory of L systems*, Academic Press, New York, 1980