computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

ADAPTED ALGORITHM OF VIRTUAL PLANTS SIMULATION BASED ON STOCHASTIC L-SYSTEM

Hai Wang [1), **Fei Hao** [2)]

[1)] Department of Computer Science, Wu Han University, Hubei, China 430072, e-mail: hkhaiwang@gmail.com
[2)] Department of Computer Science, Korea Advanced Institute of Science and Technology 373-1, Guseong-Dong, Yuseong-Gu, Daejeon 305-701, South Korea, e-mail: fhao@kaist.ac.kr

Abstract: *In this paper, the concept and characteristics of fractal are briefly introduced; The drawing principle of L-system and IFS system is discussed. Based on this discussion, we analyse virtual plant growth procedure simulated by the L-system. Further, an efficient expression that can describe the plant growth and the control of direction is defined; Inspired from proposed expression, we propose a new method based on the combination of L-system and IFS system to simulate the plant morphology. We valid and show our proposed system with high fidelity and efficiency on the simulation of plant morphology by comparison experiments. Extra experiment shows it can be applied into computer art and video game scenario design.*

Keywords: *fractal, plant simulation, IFS system.*

1. INTRODUCTION

With the rapid development of computer graphics, the simulation of different plants' morphology in nature has been a research hotspot in computer graphics. The simulation of virtual plants generated by computer has a wide application prospect in simulation of the ecological environment, virtual reality and game design, etc. The traditional approaches based on European geometric theory can only realize the simulation for some pattern with simple geometry and regulation, they can't realize the simulation for the rich natural scenery. Further, the modelling of irregular shape is also a difficult problem in terms of current virtual reality technology. The fractal geometry is a subject which focuses on the research of irregular shape, it's an important area for the simulation of natural scenery. Under the impetus of reference [1~6], L-system and IFS system have become main measures used in the modelling for virtual plant. From the point of botany, reference [7] proposed the concept of two-scale which include micro state and macro state, and established a two-scale automata model which can simulate the growth of virtual plant, at the same time, it applied a probability model which is in terms of the development process of plant terminal bud and axillaries bud, the experimental result is shown in Fig.1. By interactive deformation treatment for the branches, reference [8] provided a three-dimensional tree model which can beautify the trees' shape. The approach contained three parts: three-dimensional plant model based on interactive deformation treatment, interactive selection mechanism for branches and the deformation treatment for the selected branches. Based on aesthetic theory, this method gives an overall aesthetics control for the tree shape, and an experimental result is shown as Fig.2. From Fig.1 and Fig.2, we can easily see these two models don't concern the specific details such as leaves and flowers, so the realistic of tree model is not enough, the simulation effects remains to be further improved. In this paper, we attempt to combine the L-system and IFS system together, and propose a series of improvement approaches in order to realize the simulation of plant morphology with high fidelity.

The rest of this paper is organized as follows: section 2 introduces some basic knowledge about L-system and IFS system, and following in section 3, we propose two main improvements on the combination of L-system and IFS system. To evaluation the feasibility and efficiency of our adapted algorithm, comparison experiments are conducted in section 4; Section 5 concludes this paper.

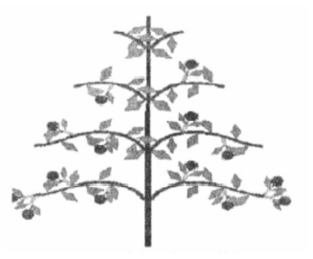


Fig. 1 – Tree simulation result with two scale model

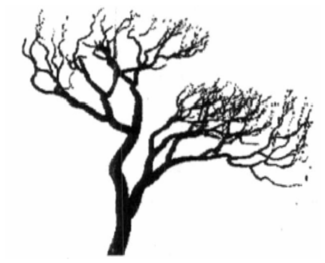


Fig. 2 – Tree simulation result with interactive deformation treatment

2. L-SYSTEM AND IFS

L-system was first put forward by American biologist Aristid Linder Mayer in 1968. Actually, L-system is a rewriting system, by empirical summary and abstract to the procedure of plant growth, we can obtain an initial state set and a rules which describe the procedure of plant growth, then replace the initial state sets with the rules, we can get a character sequence that can performance the plant topology, after geometric interpretation towards the character sequence that we get before, it will be able to generate very complex fractal images.

IFS system was first proposed by Michael F.Barnsley. He firstly generated some fractal images using a group of contraction affine transform, i.e., *contraction*, *rotation*, *translation* and other transformation towards the original graph, he got some extreme images with self-similar fractal structure, and the set of affine transformation is called as IFS.

2.1. THE REWRITING MECHANISM OF L-SYSTEM AND ITS INTERPRETATIONS

The essence of L-system is a character rewriting mechanism, the simplest L-system called 0L-system, it is a context-free grammar relationship, it can be described by a triple $< v, w, p >$, which v denotes all the characters that can be identified in the system, $w \in V^{+}$ is a non-empty string which called axiom, $p \in V$ЧV* is a finite set of production.

The axiom and production in the L-system are described by string. To produce specific image, we give the string with a specific meaning. In the L-system, we use turtle graphics to explain strings. The turtle morphology can be defined as a collection of three elements (x, y, θ), which (x, y) represents the turtle's location in the Cartesian, θ represents the direction of the turtle; In general, we give the step d and angle increment α. Two dimensional L-string and their turtle interpretations are shown in table 1:

Table 1. Turtle interpretations

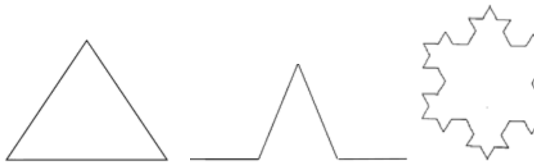
$F(d)$	Move forward one step, draw the path of motion, the turtle's next state become $(x+d*\cos\theta, y+d*\sin\theta, \theta)$
$+(\alpha)$	rotate counter clockwise by angle α, the turtle's next state will be $(x, y, \theta + \alpha)$
$-(\alpha)$	Rotate clockwise by angle α, the turtle's next state will be $(x, y, \theta - \alpha)$
[	push the current state of turtle to the stack to start a branch
]	End a branch, and pop a state from the stack and take it as current state

Take the typical Koch curve as an example; the generation rule is as follows:

$\alpha : 60°$

$w{:}F(1)—F(1)—F(1)$ The initial map

$p{:}F(d){-}{>}F(d/3)+F(d/3)—F(d/3)+F(d/3)$ Generators



Initial map Generators Iterated by 2 times

When extended into three-dimensional space, the key problem is to use three orthogonal vectors H, L and U to represent current location of the turtle, so the turtle's rotation can be described as following:

$$[H`, L`, U`] = [H, L, U]R \tag{1}$$

Which:

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

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

Based on two-dimensional L-system, we add some new symbols to describe the change direction of turtle in three-dimensional space, the new symbols and their turtle interpretations are shown in table 2:

Table 2. Symbols for rotation

$+(\sigma)$	rotate left by angle σ around U, which described by matrix $R_U(\sigma)$
$-(\sigma)$	rotate right by angle σ around U, which described by matrix $R_U(-\sigma)$
$\&(\sigma)$	rotate down by angle σ around vector L, which described by matrix $R_L(\sigma)$
$\wedge(\sigma)$	rotate up by angle σ around vector L, which described by matrix $R_L(-\sigma)$
$\backslash(\sigma)$	rotate left by angle σ around vector H, which described by matrix $R_H(\sigma)$
$/(\sigma)$	rotate right by angle σ around vector H, which described by matrix $R_H(\sigma)$

The space direction and rotation schematic figure are as Fig.3.

2.2. THE IFS SYSTEM

The Formulism of describing plant morphology used by IFS system is as following: for one plant image we have got, we can use a finite number of sub-images which are transformed from it by compression affine transformation to coverage it according to the collage theorem, and allow some overlapping between the sub-images, then we can get a set of compression affine transformation as following:

$$\{R^2 : w_1, w_2, w_3, ..., w_n\}$$
$$w_i\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r\cos\alpha & -q\sin\beta \\ r\sin\alpha & q\cos\beta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \quad (2)$$

In expression (2), character r represents the scaling factor in x direction between sub-images and source image, character q represents the scaling factor in y direction between sub-images and source image; character α and β represent the rotation angle for sub-graphs compared with the original image respectively, character e represents the displacement in x direction of sub-graph, character f represents the displacement in y direction of sub-graph. According to the area proportional of the sub-graph, we can get a probability set $\{p_1, p_2, p_3, ..., p_n\}$, which $p_i > 0$ (i=1,2,….n), and represents the probability of the corresponding transformation, p_i satisfies the following expression:

$$\sum_{i=1}^{n} p_i = 1 \quad (3)$$

Expression (2) and (3) construct a complete IFS system. When we draw graphics using the parameters of the IFS, we use random iteration algorithm, in other words, we can choose an original point（x,y）to our starting point, and then choose one affine transformation according to the corresponding probability, calculate a new point（x`,y`）and draw that point, and afterwards, we set the new point as a new original point. By repeating the above process, and iterating for many times, all the track of the point we calculated can form a new image which is similar to original image, we call this image as IFS attractor.

3. SIMULATION OF THE THREE-DIMENSIONAL TREE'S MORPHOLOGY

3.1. AN EFFECTIVE METHOD FOR THREE-DIMENSIONAL PLANT MODELING

When we extend IFS system into three-dimensional space, the basic case is the same as that in two-dimensional plane, the only difference is that the string is explained in three-dimensional space. When simulating three-dimensional plant, an important point is how to get plants' next growing point quickly. Most researchers devote their time to make the simulation result have high fidelity, but they neglect a problem that how to make the expression which represents the information of growing point simpler. In this section, we proposed a new approach which can get the next growing point more quickly.

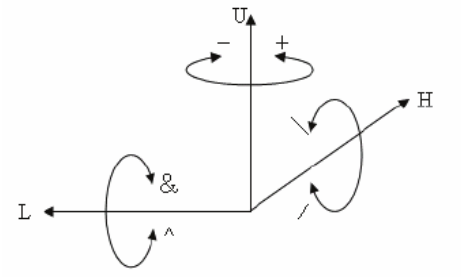


Fig. 3 – The rotation schematic diagram

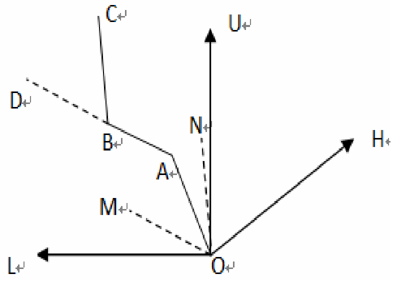


Fig. 4 – Calculating schematic diagram

The three-dimensional plant's growth procedure can be seen as a turtle's crawling procedure in space. For the problem that how to represent the turtle's crawling location, reference [9] chooses quaternion method. Although it can solve the practical problem of motion analysis and control, but it needs lots of middle steps to get the final result. Reference [10] chooses the angle between growth direction and X axis, Y axis, Z axis as transform parameters, but for design and implementation for algorithm, it needs many additional parameters. Here, we adopt a collection of six elements $(x, y, z, \alpha, \beta, \gamma)$ to represent current status of the turtle. In expression, (x, y, z) represents the location of the turtle's current status, (α, β, γ) represents the turtle's current direction, it equals to the direction of vector (α, β, γ) , and when in actual calculating procedure, it meets $\sqrt{\alpha^2 + \beta^2 + \gamma^2} = d$ (step length). As depicted in Fig.3, suppose $A(x, y, z)$ as the location of turtle's last status. $B(x`, y`, z`)$ represents the location of turtle's current status, $C(x_1, y_1, z_1)$ is the next location which need to be calculated. According to the explanation of the turtle's map, when the turtle craws to point B and don't rotate yet, the current turtle's crawling direction is equal to the direction of vector $\overrightarrow{AB}$, that is $(\alpha, \beta, \gamma) = (x`-x, y`-y, z`-z)$.

To calculate the next location $C(x_1, y_1, z_1)$, we can extend the segment AB to point $\mathrm{D}(x_2, y_2, z_2)$ and let $AB = BD = d$. Suppose we face $+(\sigma)$ in string, it means that taking B as non-fixed point, counter clockwise rotate point D around U axis by angle σ, then point C coincides with point D. So, the coordinate value of point C is calculated as following:

(1) Take origin as the start point of vector $\vec{a}$, let $\vec{a} = (x`-x, y`-y, z`-z)$ , suppose the end point of vector $\vec{a}$ is M, obviously, the coordinate value of point M satisfies: $(x`-x, y`-y, z`-z) = (\alpha, \beta, \gamma)$;

(2) We can take origin as non-fixed point, rotate point M by $+(\sigma)$, we call the new point as N, and so the coordinate value of N point can be calculated by the rotation matrix. The specific calculating methods are as following:

$$\begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix}^T = \begin{pmatrix} x`-x \\ y`-y \\ z`-z \end{pmatrix}^T \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}^T \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{4}$$

After rotation, the turtle's direction of current status have been changed, the new direction meets:

$$(\alpha, \beta, \gamma) = (x_3, y_3, z_3) \tag{5}$$

(3) Translate vector $\vec{b}$ which starts with origin and ends with point N, make point B as the new start point of vector $\vec{b}$, then we can see that point C will be the end point of vector $\vec{b}$, so the coordinate value of point C can be calculated by using following formula:

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}^T = \begin{pmatrix} x` \\ y` \\ z` \end{pmatrix}^T + \begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix}^T = \begin{pmatrix} x` \\ y` \\ z` \end{pmatrix}^T + \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}^T \tag{6}$$

From formula (6), we can see that: the location of next status can be got by adding that of current status and the direction of current status. At the same time, using a vector to represent the direction, we can easily calculate the direction of turtle's status after rotation. Compared with the approaches used in reference [9] and [10], the approach used in this paper only uses the addition and subtraction of matrix and little multiplication, it avoids many complicated calculations. The whole method is simple and practical, and has a high operability.

In the procedure of crawling, if the turtle rotate for many times, what we should do is only to repeat the step (2), but remember that: in every time we calculate, we should change the rotation matrix to the corresponding matrix.

3.2. THE PROPOSED ALGORITHM

Tree is a typical kind of plant, their appearance varies from one to another, but we can find that all trees have the same growth mechanism, namely most trees have two parts:

Spindle: branches which have child branches, all branches composite the plant topology.

Terminal: tree's terminals include leaves or flowers instead of child branches in general.

So the three-dimensional simulation for plant mainly includes two parts: the simulation for spindle and that for terminal.

3.2.1. THE SIMULATION FOR SPINDLE

Parameter L-system has advantages in the simulation for tree's growth pattern and topology; it can describe the random growth mechanism and topology of the branch freely and conveniently. By controlling and changing parameters in fractal unit and production rules, we can realize the modelling for complicated three-dimensional topology, an example is shown as following:

$$\begin{aligned}&w:(w0)F(d)\\&p:(w0)F(d)->(W)[F(d)][M(t1*d)[/(B1)\\&+(a1)F(r1*d)][\backslash(B1).(a1)F(r1*d)][\backslash(B2)+\\&(a1)F(r1*d)]][M(t2*d)[\backslash(B1)+(a2)F(r2*d)]\\&[\backslash(B1).(a2)F(r2*d)][/(B)+(a2)F(r2*d)]]\end{aligned}\tag{6}$$

$r1, r2$: represents the branch's scaling ratio of the first, second branch node respectively.

$a1, a2$: represents the angle between child branch and father branch of the first, second branch node respectively.

$B1, B2$: represents the rotation angle between child branch and father branch's vertical of the first, second branch node respectively.

$t1, t2$: represents the ratio between child branch's length and father branch's length of the first, second branch node respectively.

The model based on this algorithm has much advantage on the control for the tree's specific details. But once parameters assigned value, the tree's topology are the same, but in real nature, even the same kind tree also have some differences in detailed morphology because of the different habitat and different inner gene. To get a more extensive effect, based on the above algorithm, we introduce random elements, so we contribute some improvements for the above parameter L-system as following:

We can set several different values for each parameter, and for each value, set a probability factor p which means the probability of being chosen. When painting, choose one value each time. All the probability factors for one parameter meet $\sum_{i=1}^{n} p_i = 1$. For example, as shown in expression (7) and (8), we can set three values to choose for radius scaling ratio $r1$, when painting, choose one according to the corresponding probability factor.

$$r1 \in \{\sin(\pi/4), \sin(\pi/3), \sin(\pi/6)\} \tag{7}$$

$$p[\sin(\pi/4)] + p[\sin(\pi/3)] + p[\sin(\pi/6)] = 1 \tag{8}$$

Likewise, when painting new branch, the ratio between the radius of child branch and that of father branch is not a fixed constant, but a value which fluctuates within a certain range, so the tree's morphology would be more close to natural law.

There is also another deficiency in above model mechanism, namely, in the same branch: the bottom of the branch is larger than the top. In order to simulate the branch better, we use a truncated cone to simulate branches, and we can also set $r3$ as the ratio between the radius of upper face and that of bottom face in the truncated cone, the processing method for $r3$ is similar to that for $r1$.

Actually, in the process of plant growth, some plant's topology structure would change when it becomes closer to the terminal mud, so when painting, we can add a judgment for iteration times, when the times reach one given number, the topological rules would be changed.

3.2.2. THE SIMULATION OF TERMINAL

In the aspect of simulating plant's morphology, IFS system has the following advantages: the generated graphics is coloured; the shade gradually changes and graphics has rich texture. For those reasons, IFS system is fit to simulate the graphics with rich texture such as tree's leaves and flowers. To integrate the L-system with IFS system to simulate plant's morphology, we encapsulate the IFS algorithm as a function, with different parameters and set of compression affine transformation, we can paint leaves with different location, different size and different morphology. When simulating, we don't paint a truncated cone to simulate the terminal, instead, we call the encapsulated IFS function and assign corresponding value for each parameter, then paint leaves for the plant's terminal.

The combination for IFS system and L-system is something like "building block". On one side, use the generated rules and parallel rewriting mechanism of L-system as the method for "building block", on another side, use the graphics with rich texture generated by IFS system as "block". By this improved algorithm, we can simulate many kinds of plant's morphology with high fidelity.

3.2.3. THE IMPLEMENTATION FOR PROPOSED ALGORITHM

According to the analyses above, we can use two modules to implement our proposed algorithm, one is character parallel rewriting module, and another is turtle map explanation module. The algorithm flows of the two modules are shown in fig.5 and fig.6 respectively.

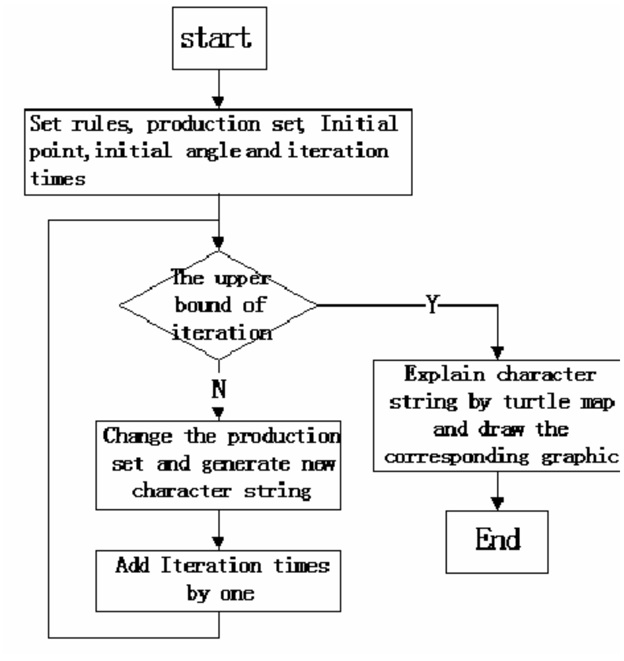

start
Set rules, production set, Initial point, initial angle and iteration times
The upper bound of iteration
Y
N
Change the production set and generate new character string
Add Iteration times by one
Explain character string by turtle map and draw the corresponding graphic
End

Fig. 5 – The algorithm for character parallel rewriting module

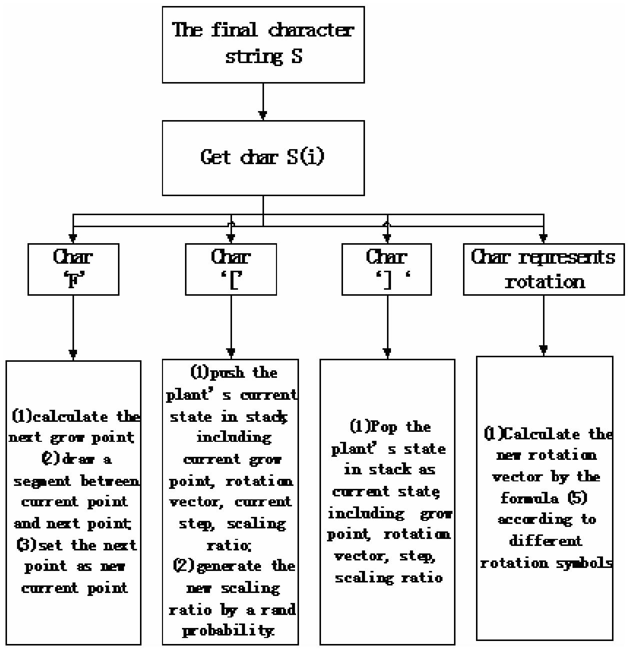

The final character string S
Get char S(i)
Char 'F'
Char '['
Char '] '
Char represents rotation
(1)calculate the next grow point (2)draw a segment between current point and next point; (3)set the next point as new current point
(1)push the plant's current state in stack, including current grow point, rotation vector, current step, scaling ratio; (2)generate the new scaling ratio by a rand probability.
(1)Pop the plant's state in stack as current state, including grow point, rotation vector, step, scaling ratio
(1)Calculate the new rotation vector by the formula (5) according to different rotation symbols

Fig. 6 – The algorithm for turtle map explanation module

To implement the above two modules, we use three key functions. Before show the three functions, we define the following basic characters:

Table w: storage the initial map;

Table p: storage the generators;

Stack: the stack we used in function;

Point pt: represent the current grow point;

Step: the initial step;

Scale: the scale factor to regulate plant's growing morphology;

N: iteration times;

1. The main function for the whole system

```
1.  int i=0, char v=w.at(i);
2. if( i >w.length())
3. {
4.exit();
5. }else {
6. switch (v) {
7. case  'F': L-rule(N);
8. break();
9. case '+': calculate new rotation vector; break();
10.case '-': calculate new rotation vector;  break();
11.case'&': calculate new rotation vector; break();
12.case '^': calculate new rotation vector; break();
13. case '\': calculate new rotation vector; break();
14. case '/': calculate new rotation vector; break();
15.  }
16. i=i+1;
17. }
```

2. The iterated function L-rule(n)

```
1. if(n==1)
2. {    draw();
3.      exit();  }
4. else if (n==0)
5. {      leaf(); //call  the encapsulated IFS function
6.int j=0;
7. }else {
8.int j=0; }
9. if(j>p.length())
10. exit() ;
11. else {
12.  char t=p.at(j);
13. switch (t){
14. case 'F': Factor=(Rand()%10)/55;
15.           step=step*Factor;
16.           n=n-1;
17.           L-rule (n-1);
18.           break();
19. case  '+': calculate new rotation vector; break();
20. case  '-': calculate new rotation vector;  break();
21. case '&': calculate new rotation vector; break();
22. case  '^': calculate new rotation vector; break();
23. case  '\': calculate new rotation vector; break();
24. case  '/': calculate new rotation vector; break();
25. case '[':
26.stack.push(currentstate);
27.           Factor=(Rand()%10)/55;
28.           step=step*Factor;
29.           break();
30.  case =']':currentstate=stack.pop(); break();
31.  }
32.  j=j+1;
33. }
```

3. Draw function

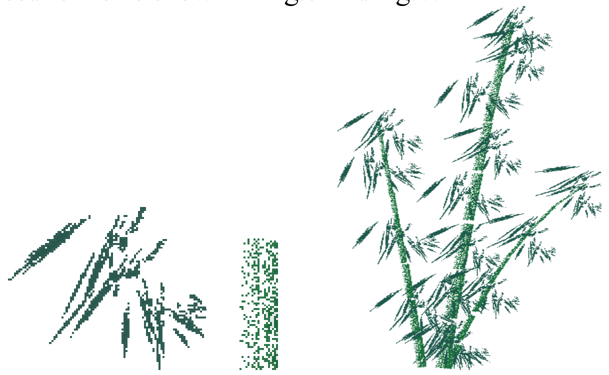
```
1. draw initial map;
2. calculate next growing point pn;
3.      pt=pn;
4.  calculate new rotation vector;
5.   exit() ;
```

As mentioned in 3.2.2, the encapsulated IFS function is a basic function for IFS system, here we don't give its specific implementation.

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1. EXPERIMENTAL RESULTS

We implement our proposed improved algorithm with Visual C++ and OpenGL library, in the process of simulation, considering the random element of the diagram parameters involved in L-system and IFS system, we carried out a large number of plant simulation experiments, some typical experimental results are as shown in fig.7 and fig.8.



(a) bamboo's leaves (b) bamboo's node (c)Bamboo

Fig. 7 – The simulation result of bamboo

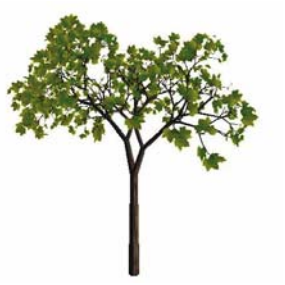


Fig. 8 – The simulation result of tree

In this paper, we redefine the parameters in expression(1): $a = r\cos\alpha, b = -q\sin\beta, c = r\sin\alpha, d = q\cos\beta$, the data used in this paper are as following:

Fig.7(a)'s affine transform coefficient are shown in table 3.

Table 3. Fig.7 (a)'s affine transform coefficient

w	a	b	c	d	e	f	p
1	0.29	0.4	-0.4	0.0	0.28	0.44	0.25
2	0.33	-0.34	0.39	0.4	0.41	0.0	0.25
3	0.42	0.0	0.0	0.63	0.29	0.36	0.25
4	0.61	0.0	0.0	0.61	0.19	0.23	0.25

Fig.7(b)'s affine transform coefficient are shown in table 4.

Table 4. Fig.7 (b)'s affine transform coefficient

w	a	b	c	d	e	f	p
1	0.5	0.5	0.0	0.0	0.0	0.0	0.15
2	0.5	0.5	0.0	0.0	50	0.0	0.35
3	0.5	0.5	0.0	0.0	0	50	0.35
4	0.5	0.5	0.0	0.0	50	50	0.15

Fig.8's affine transform coefficients are shown in table 5.

Table 5. Fig.8's affine transform coefficient

w	a	b	c	d	e	f	p
1	0.0	0.0	0.0	0.16	0.0	0.0	0.01
2	0.85	0.04	-0.04	0.85	0.0	1.6	0.85
3	0.2	-0.26	0.23	0.22	0.0	1.6	0.07
4	-0.15	0.28	0.26	0.24	0.0	0.44	0.07

Nowadays, Fractal is becoming more and more important to computer art and video game scenario design. For example, the natural landscape (Shown in Fig.9) is a key application using proposed approach, and the affine transform coefficient for this figure are given in appendix.

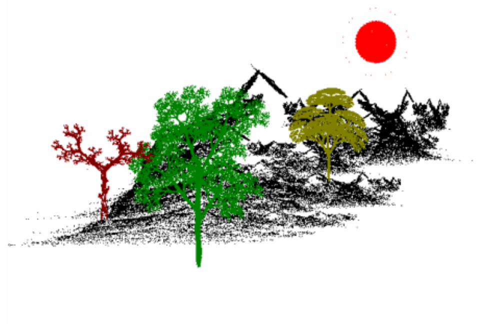


Fig. 9 – Simulation result of landscape

4.2. COMPARISON EXPERIMENTS

The algorithms proposed in reference [9] and [10] are typical algorithms which use L system to simulate plant's morphology. With the same experimental environment mentioned in section 4.1, we implement the algorithms used in reference [9] and [10], and some experimental results are shown in fig.10 and fig.11.

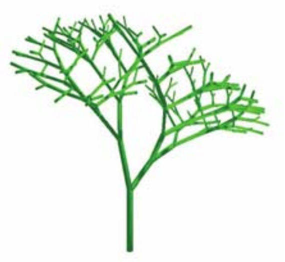


Fig. 10 – Simulation result of trunk

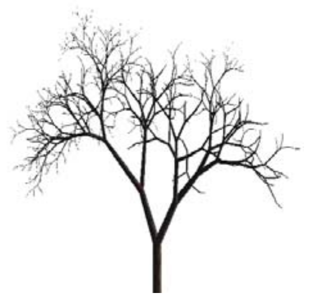


Fig. 11 – Simulation result of tree

As can be seen from the comparison results, the plant can be simulated more accuracy using our proposed approach.

4.3 RESULTS DISCUSSION

Compared with algorithms proposed in reference [9] and [10], our algorithms mainly present two improvements.

(1) The fidelity of plant simulation result

From the experimental results shown in section 4.1 and the comparison experimental result shown in section 4.2, obviously we can find that our proposed algorithm can simulate plant's morphology with higher fidelity. There is an important extra point we must mention especially, i.e., from fig.9, we can see that our algorithm also have an extensive application prospect in simulation of natural scenery.

(2) The complexity for controlling plant's growth process

As mentioned in section 3.1, we give a simpler expression which can calculate the state of next growing point than that proposed in reference [9] and [10]. Here, we give the other two schematic diagrams which used to calculate the next grow point. Fig.12 (a) and (b) are the schematic diagrams used in reference [9] and reference [10] respectively.

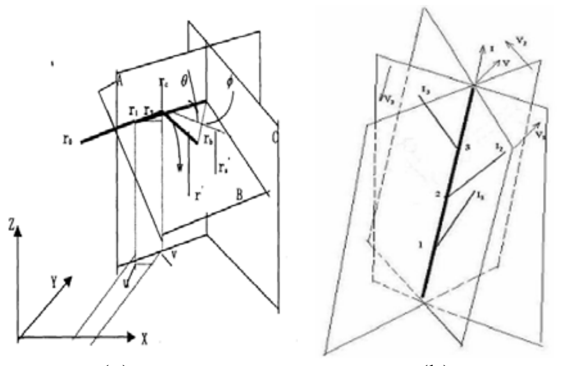


Fig. 12 – Schematic diagram for plant growing direction controlling

From fig.12, we can see that our formula is simple and effective to control plant growing direction.

When conducting experiments, we record a series of rendering time for simulation result shown fig.8 and that for fig.11. The relationship between total rendering time and the rendered amount is shown in fig.13.

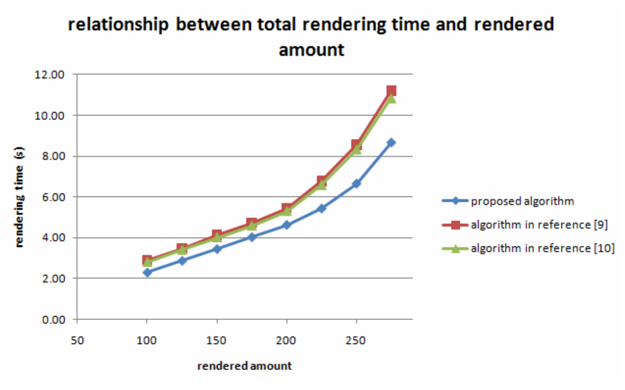


Fig. 13 – Relationship between total rendering time and rendered amount

At the same time, we give the relationship between single average rendering time and the rendered amount in fig.14.

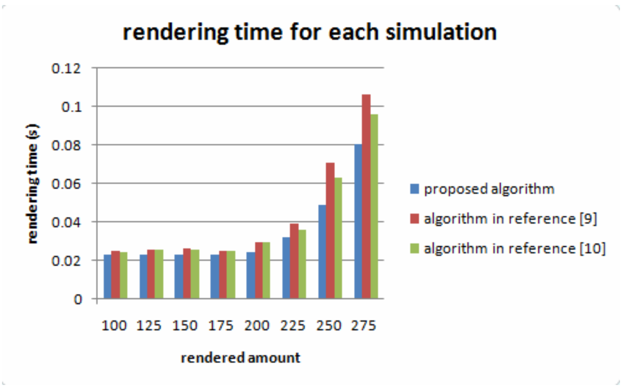


Fig. 14 – Relationship between each single rendering time and rendered amount

From fig.13 and fig.14, we can see that our proposed approach mainly has two improvements in rendering time:

(1) we can simulate a plant with higher fidelity using less time than that of algorithm proposed in reference [9] and [10], and the efficiency increases 13 % approximately;

(2) when the rendered amount is large, using our proposed algorithm, the average stimulation time for each object increases more slowly than that of algorithm proposed in reference [9] and [10], and this improvement has obvious advantage in large-scale simulation for natural landscape.

5. CONCLUSIONS

This paper aims to improve the traditional L-system, gives a simpler method to calculate plant's next growing point. Based on parametric L-system, we add the stochastic element, and combine stochastic L-system with parameter L-system together to make the simulation results have higher fidelity. At last, we use the IFS algorithm to generate the leaves, so that the graphics painted by L-system would be closer to the plant in nature. Experimental results show that our proposed approach can simulate virtual plants with high fidelity using less rendering time. But in this paper, we don't consider the influence of gravity on the branches' morphology [11~13], we also don't consider the environment (such as wind) around the plant in nature, so our further work is to build a good dynamic model for plant's branches and to realize the branches' dynamic flicker in wind.

Appendix

The affine transform coefficient for hill

w	*a*	*b*	*c*	*d*	*e*	*f*	*p*
1	0.08	-0.031	0.084	0.0306	5.17	7.97	0.03
2	0.0801	0.0212	-0.08	0.0212	6.62	9.4	0.025
3	0.75	0	0	0.53	-0.375	1.106	0.22
4	0.943	0	0	0.474	-1.98	-0.65	0.245
5	-0.402	0	0	0.402	15.513	4.588	0.21
6	0.217	-0.052	0.075	0.15	3	5.741	0.07
7	0.262	-1.105	0.114	0.241	-0.473	3.045	0.1
8	0.22	0	0	0.43	14.6	4.286	0.1

The affine transform coefficient for green tree

w	*a*	*b*	*c*	*d*	*e*	*f*	*p*
1	0.05	0	0	0.6	0	0	0.1
2	0.05	0	0	-0.5	0	0.8	0.14
3	0.4596	-0.3214	0.3857	0.383	0	0.48	0.2
4	0.4698	-0.1539	0.171	0.4229	0	0.88	0.18
5	0.433	0.275	-0.25	0.4763	0	0.8	0.18
6	0.4505	0.2294	-0.3155	0.3277	0	0.4	0.2

The affine transform coefficient for palm tree

w	*a*	*b*	*c*	*d*	*e*	*f*	*p*
1	0.195	-0.488	0.344	0.433	0.4431	0.2452	0.2
2	0.462	0.414	-0.252	0.361	0.2511	0.5692	0.2
3	-0.058	-0.07	0.453	-0.11	0.5976	0.0969	0.2
4	-0.035	0.07	-0.469	-0.022	0.4884	0.5069	0.2
5	-0.637	0	0	0.501	0.8562	0.2513	0.2

The affine transform coefficient for gray tree

w	*a*	*b*	*c*	*d*	*e*	*f*	*p*
1	-0.04	0	0.23	-0.65	-0.08	0.26	0.25
2	0.61	0	0	0.31	0.07	2.5	0.25
3	0.65	0.29	-0.3	0.48	0.54	0.39	0.25
4	0.64	-0.3	0.16	0.56	-0.56	0.4	0.25

The affine transform coefficient for sun

w	*a*	*b*	*c*	*d*	*e*	*f*	*p*
1	0.15596	0.98776	-0.98776	0.15596	-0.779	9.124	0.9866
2	0.04428	0	0	0.04116	0.641	4.829	0.0032
3	0.05566	0	0	0.04527	0.998	4.779	0.0029
4	0.1154	0	0	0.05094	1.428	4.761	0.0036
5	0.27142	0	0	0.04923	2.38	4.781	0.0037

6. REFERENCES

[1] A. Lindenmayer. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs, *Journal of Theoretical Biology*. 1968, 3, p.300-315.

[2] M. F. Barnsley, S. Demko. Iterated function systems and the global construction of fractals, *Proc. Roy. Soc. London Ser.* 1985, pp. A.399, 243–275.

[3] M. F. Barnsley. Fractal functions and interpolation, *Journal Constr. Approx*. 1986, pp.303–329.

[4] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, James Hanan. Development models of herbaceous plants for computer imagery purposes, *SIGGRAPH*, 1988, pp.141-150.

[5] Prusinkiewicz P., Hammely M. S., Mjolsness E., etc. Animation of plant development, *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. Anaheim, CA, 1993, pp.351–360.

[6] Prusinkiewicz P., Lindenmayer A. *The algorithmic beauty of plants*. New York: Springer-Verlag, 1990, pp. 40–50.

[7] Zhao Xing, de Reffye Philippe, Xiong Fanlun, etc. Dual Scale Automaton Model for Virtual Plant Development, *Chinese Journal of Computers*, 2001, 24 (6), pp.608–615.

[8] Pan Yun-He, Mao Wei-Qiang. Research on Interactive Morphing Based 3D Modeling of Tree, *Journal of Computer Aided Design & Computer Graphics*, 2001, 13 (11), pp.1035–1042.

[9] Ke Guan. The Plant Modeling Research Based on Improved 3D L-System and Quaternion, *Proceedings of the 2008 IEEE International Conference on Information and Automation, ICIA*, 2008, pp. 828-833.

[10] Li Feng, Li Wang. Algorithm of 3D Fractal Plants Based on L-system and Its Implementation, *Journal Computer Simulation*, 2005, 22 (11), p. 205–208.

[11] Hitoshi Kanda, Jun Ohya. Efficient, realistic method for animating dynamic behaviors of 3D botanical trees, *Proceeding of the 2003 IEEE International Conference on Multimedia and Expo, IEEE Computer Society*, 2003, 1(7), p. 89-92.

[12] J. Beaudoin, J. Keyser. Simulation levels of detail for plant motion, *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004, p. 297-304.

[13] M. Shinya. Fourier stochastic motion – motion under the influence of wind, *Proceedings Eurographics'92*, 1992, 11(3), p. 119-128.

***Hai Wang**, born in 1988 in Hubei, China. With high interest in computer science, in 2007 he entered Wu Han University, where he is studying at department of computer science. In his B.S. period, he does some projects with his mentor, mainly about distributed simulation,digital image processing. Now still greatly interested in image compression with IFS.*

***Fei Hao** received the B.S. and M.S. degrees in school of Mathematics and Computer Engineering from Xihua University, Chengdu, China, in 2005 and 2008, respectively. He is currently working toward the PHD degree in the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. He has published 12 research papers in International and National Journals as well as conferences.*

His research interests include intelligent information processing, social computing and time series data mining.