Broad-leaf Virtual Plant

Wenhui Li, Wu Guo, Guanghui Feng, Yu Meng, Ming Li[+]
Key Laboratory of Symbolic Computation and Knowledge Engineer of Ministry of Education,
College of Computer Science and Technology, Jilin University, 130012, Changchun, China
[+]JiLin Province Economic Information Center, 130012, Changchun, China
E-mail:guowurita@gmail.com

***Abstract*-Virtual plant plays key roles in Virtual Reality. And efficient and realistic animation constitutes important foundation of it. This paper proposes a new modeling and animation method of broad leaf plant. It shows how to use a physically based animation method to realistically model the 3D foliage of plant with thickness information and to emulate the movements of stem and foliage in wind. In order to achieve fast collision detection we propose a suitable non-hierarchical spatial decomposition structure combined with oriented bounding boxes (OBB) for deforming broad leaf plant that can be constructed in a pre-processing phase, and then updated efficiently during the animation.**

I. Introduction

An extensive amount of work has already been carried out in the area of plant. Lindenmayer and Prusinkeiwicz built the tree model by using L-Systems method initially, which is one of the parametric approaches used to make tree models [1,2]. Oppenheimer used fractals and the notion of fractal self-similarity for modeling plants in [3]. Weber et al. proposed one method based on geometrical observables to create and render trees [4]. Wong divided leaves of plant into several very short segments in order to represent the softness of leaves in quaternion space [5]. Sakaguchi involved gathering 3D volumetric data to create a realistic model [6]. Ono described another segment based method in which braches are connected by springs [7]. Most research above can not show the realistic shape of each leave in the wind. To solve the unrealistic movements of foliage problem, we propose a method for rendering and simulating the blades of broad leaf in realistic manner.

II. Broad Leaf Plant Model

Real plants have highly complex structures, leading to complicated and expensive models. However, we do not model the true physics of the movements, but opt for a method that can represent the realistic movements of plant. For the purpose of this paper, a greatly simplified model is used. We model the branch and leaf of plant by using uniform structure.

The stem or branch model of plant is composed of several little cylinders or prism primitives called segments with equal length and different radius, which can generate rotation and translation acted by the internal forces and external forces. Modeling a leaf of plant consists of two phases. The first phase is mid-rid and parallel venation model, which is similar to the stems model with cubic tubes instead (Fig. 1a). This model can greatly simplify the expensive computation in most node point based method. The second phase is the constraint force model in the latitude direction of a broad leaf. We have been modeled this force with a spring-looking between the ends of each pair of mid-rib tubes and parallel venation tubes (Fig. 1b).

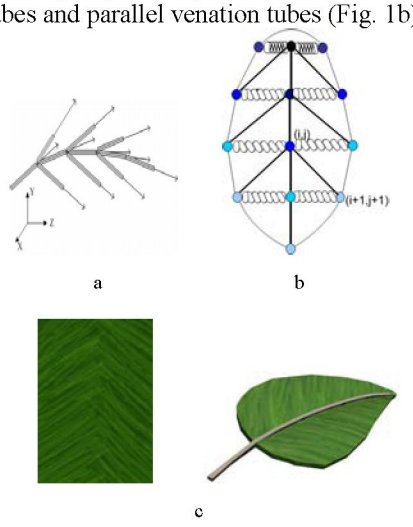


Fig. 1. a. A leaf model consists of some mid-rib tubes and some parallel venation tubes. b. The constraint force model of leaf. c. Leaf texture and leaf model adopted in our experiments.

This kind of mold is more suitable for some ordinary leaf shapes of plant, such as oblong, elliptic, rhombic and spoon shape, which have parallel venation in common. It can be seen that there is one mid-rib in the middle of leaf blade in the longitudinal direction. The constraint in the latitudinal direction may guarantee the movement of leaf blades with the wind and avoid the abnormal phenomenon with tearing and over-bending of leaves in the movement.

III. Animation of Broad Leaf Plant Model

The animation of broad leaf plant swaying in wind is mainly created by the forces acting on the stems or branches, which make meshes attached to them rotation and translation to form a variety of postures of plants. The overall animation process is described as follow.

At each time step,

1) Clear the forces on all segments, including stems and leaf tubes.

2) Traverse the segment list in the order from leaf to root segment and calculate all internal and external forces and generating a new rotation for each segment.

3) Detect self-collision and perform collision response

4) Traverse the segment list again in reverse order and calculate the new rotation of each segment by adjusting segment structure for translation or restriction. Use this new rotation to compute the final absolute transformation matrix of each segment.

5) Render and display the image of plant at this time.

We will discuss the detail of force calculations of stem and leaf tube of plant and restriction of leaf tube as follow.

A. Dynamics of Stem and Leaf Tube

The movement of each stem and leaf cubic tube is calculated in a Cartesian coordinate system. Matrices adopted are considered as a very convenient and simple method to calculate rotations and translations in the three dimensional space. Segment is considered as the smallest element of the plant that can independently be animated. We can individually model the motion of each segment for determining the dynamics of the segments. So the problem is formulated as follows: assuming we know the rotational movement $\vec{R}^t$, velocity $\vec{V}_\alpha^t$ and resultant force $\vec{F}^{stem}$ acting on a segment at time t, we want to compute the new rotation of the segment $\vec{R}^{\tau+\Delta\tau}$ by

$$\vec{R}^{t+\Delta t} = \vec{R}^t + \Delta t \bullet (\vec{V}_\alpha^{t+\Delta t} + \Delta t \bullet \frac{3}{ml^2} \bullet \vec{\tau}^{t+\Delta t}) \quad (1)$$

where m is the mass of a segment , and l is the length of a segment, $\vec{\tau}^t$ is the torque.

The first-order Euler's method although very simple is subject to numerical instability. We instead use the Verlet integration method which is more stable and less time-consuming [8]. Now all that is left is to calculate $\vec{F}^{stem}$ for a segment, which is the sum of all the different types of forces acting on it. Note that when looking at each segment in isolation, we only consider rotational movement. Any translational movement of a segment is due to a movement away from its parent segment.

Weight force, $\vec{F}_G$ due to gravity (weight) intuitively is an external force. Note that the mass of stem segment is different from that of leaf tube.

Restoration force, $\vec{F}_R$ is an internal force which always acts to restore the initial orientation of the segment. To provide a robust and flexible technique to animate realistic plant movements, it is necessary to include a variable that can ultimately determine the behavior of the plant.

$$\vec{F}_R = \mathrm{K}_r r^2 (\vec{R} \times \vec{Z}) \quad (2)$$

K_r is a user defined constant, representing the rigidity of the segments, r is the cross-section radius. $\vec{R}$ is the current rotation vector of the segment, and $\vec{Z}$ is a unit vector pointing along the local Z-axis. By introducing the notion of rigidity, it is not only to control the general behavior of the plant efficiently but to reduce the complexity of the process of force analysis.

Axial damping force, $\vec{F}_D$ is an internal force which opposes the angular velocity of the segment. Its calculation is similar to the restoration force.

Transfer force, $\vec{F}_T$ is defined as an external force that is propagated from a segment, back to its parent. For example, when wind is exerted to a leaf or a branch ($i+1$th segment), some of this force is transferred to the branch it is attached, causing ith segment to bend more than it would if the leaf or the branch did not exist. In our experimentation, we have taken a user defined fraction of the external forces acting on a segment, and apply to its parent, which is accordant with the order of nature.

$$\vec{F}_T^i = \mathrm{K}_t \bullet \mathrm{K}_r \sum_{i=1}^{n} (\vec{F}_W^i + \vec{F}_G^i + \vec{F}_T^{i+1}) \quad (3)$$

K_t is user defined constants representing transitivity, n is the number of child segments. $\vec{F}_W^i$ and $\vec{F}_G^i$ are the wind force and weight of the ith child, and $\vec{F}_T^{i+1}$ is transfer force of the i+1th child, after being transformed to the local space of ith segment. The value of the segment the apex of leaf attached is 0.

Wind force, $\vec{F}_W$ is an external force and can be defined by users. Then the internal and external force vectors are summed to give the final force acting on a segment.

$$\vec{F}^{stem} = \vec{F}_W^{\,t} + \vec{F}_G^t + \vec{F}_R^{\,t} + \vec{F}_D^{\,t} + \vec{F}_T^{\,t} \quad (4)$$

B. Constraint Force of Leaf Tube

The movement of leaves is determined by the dynamics of each cubic tube of leaf model in the mesh. In order to keep the shape of leaf acted by the external force, a kind of constraint force is introduced. It is not a real force. But a very firm restriction that we assume that the rotation angle α must be between $\alpha_{\min}$ and $\alpha_{\max}$. That is to say, if the angle between the mid-rib rotation vector and the parallel venation vector after rotation is more than $\alpha_{\min}$ or less than $\alpha_{\max}$, we must adjust the angle to avoid the leaf with tearing and corrugation.

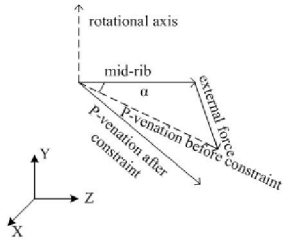


Fig 2. Adjustment process of a parallel venation acted by constraint force

The new parallel venation vector can be computed by following algorithm.

Step 1. Think about the local Cartesian coordinate of mid-rid connected with. The origin of this coordinate system is translated into the base of a new parallel venation.

Step 2. Calculate the angle α between mid-rib and temporary parallel venation generated in last section by dot product.

Step 3. If rational angle α is not more than $\alpha_{\min}$ or less than $\alpha_{\max}$, go to step 6.

Step 4. Generate the rotational axis by cross product of mid-rib and temporary parallel venation.

Step 5. Create the new parallel venation vector by the rational axis and the assigned angle, and delete the temporary parallel venation vector.

Step 6. The parallel venation is the resultant parallel venation.

The dynamic generation of the cubic tube provides the option to easily alter the number of tubes in longitudinal and constraint force in latitudinal direction. The more tubes we are used to represent a model, the finer the movements that can be achieved, and the more expensive computation it will take.

IV. Collision Detection

In order to generate naturalistic animation of broad leaf plant we need to detect whether a self-collision occurred or not and perform a logical collision response. The original method tests each pair of triangles but this is time consuming. A fast and accurate algorithm whose complexity is independent of object's complexity is needed.

A. Build Collision Model

We propose to use the non-hierarchical structure of uniform scene spatial decomposition combined with the approach of spatially partitioned oriented bounding box (OBB) volumes to solve the problem in collision detection of the broad leaf plant. This is done by partition the whole virtual space into homogeneous regions with equal volume, we just perform overlap test between the OBBs that hold the same region or adjacent region, and this might be an effective choice to reduce the number of OBBs referred to intersection tests

An OBB [9] is an arbitrarily oriented minimum rectangle bounding the object in 3-space, it is defined by a centre and three axes forming the base of its coordinate system. We define each leaf, each segment of a stem as an independent primitive, and then build the bounding box for each primitive including the information of the relationship between adjacent primitives according to the hierarchical structures of the plant. We can use all the vertices associated with the primitive to compute an OBB.

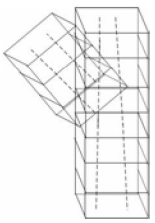


Fig.3. Subdivisions along OBBs of stems, dashed line denotes stems

First, the three axes are easily computed by obtaining the covariance matrix S associated with the subset of vertices and then obtaining its eigenvectors according to Principal Component Analysis (PCA) [10]. Second, the extents and center are computed by the maximum and minimum projections of each vertex along 3 axes. The covariance matrix S is given by

$$s = \frac{1}{n}\sum_{i=0}^{n}(X_i - \overline{X})^t(X_i - \overline{X}) \qquad (5)$$

where Xi = (xi,yi,zi)are coordinates of vertices of a primitive.

In order to sufficiently reduce the number of triangles fall into test, we subdivide the OBB into several parts. This is done by partitioning the main axis of the OBB and then placing orthogonal planes at each of the partitions. The vertices attached to the primitive are separated in sub-lists given by the space subdivision obtained by planes orthogonal to the main axis and located in each one of the subdivisions along the axis.

During simulation, once two OBBs are found to overlap each other, we only need to project it onto the main axis of each OBB to find out to which subdivision it belongs. Then intersection tests will be performed only against the triangles falling in certain subspaces of the OBBs.

B. Collision Detection and Collision Response

The potential collisions: the general case of collision detection is the one involving a plant object and another object of the scene; a particular case is the case of self-collisions, i.e. collisions of the deformable plant object with itself. We mainly discuss the case of self-collision which involves detection between two segments of a stem, detection between two leaves, detection between a leaf and a segment, detection between a leaf and itself.

In the case of self-collision detection of the leaf and stem, another optimization based on the surface curvature [11] has been implemented. The optimization has the following property: when a given zone has a sufficiently "low curvature", it can't self-intersect. The curvature of a zone denotes by the set of normals of the triangles belonging to the zone, and then a cone which includes all these normals is computed. The cone angle α gives a sufficient priori condition: if $\alpha < \pi$, the zone cannot self-intersect. Therefore if the given cone is comparatively flat, no self- collision tests are required at all. By propriety restricting the rotation angle between segments, the zone of each lead and each stem may completely fit the condition above, so there is no need to test whether a leaf or stem intersect itself.

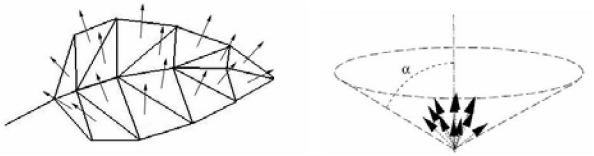


Fig.4. Normals of triangles on a leaf surface and the cone including all these normals

Because of the constraint relationship among linked segments, so collision detections between stem segments, leaves, leaf and stem segment, and some primitives with the relationship of paternity do not intersect at all thanks to the constraint force on rotation angle, and we need not test these pairs.

The most intuitive way to handle collisions is inserting a very stiff spring [10]. Thus, when a collision of a pair of triangles is detected, a spring force is temporary applied equally and in opposite directions between the segments contain the triangles (Eq.6), where K is a spring constant controlling the stiffness of the spring, D is the interpenetration depth between segments. The functional form goes to infinity as the interpenetration depth D approaches some small value.

$$F_s = K/D \quad (6)$$

C. *Update during Simulation*

The axes of the OBB can be easily updated by applying the matrix transformation, as a result of the movement of each primitive relative to his parent carrying out by rotation, when the movement happens we just apply a rotation matrix to the OBBs of the primitive and its descendant. Furthermore, the rotation transformation also causes a translation transformation on its descendant，so we also update the translation matrixes of the OBBs belong to the primitive's descendants.

The only problem now would be to update the size of the OBB. Since the stem segment does not change its size during movement, we do not update the size of its OBB. Each leaf is composed of several segments, therefore the movement of a leaf segment may produce a deformation of the leaf so do the OBB belongs to the leaf primitive. As a result of restricting the angles between adjacent segments, we can compute the maximum transformation of the leaf along the local axes, so that we obtain a slightly bigger OBB whose size is the maximum extent along each axis. Therefore this initial value for the size of OBB can be used during simulation time without any need to be updated. This approach provides a more comprehensive way to trade-off accuracy for computation time.

V. Experimental Results

In our experiments, a broad leaf plant model was used to animate the behavior of the plants in nature (shown in Fig. 5). The simulation is implemented by a computer with Pentium IV-2.8GHz CPU, 512MB DDR RAM and ATI 9600 128MB.

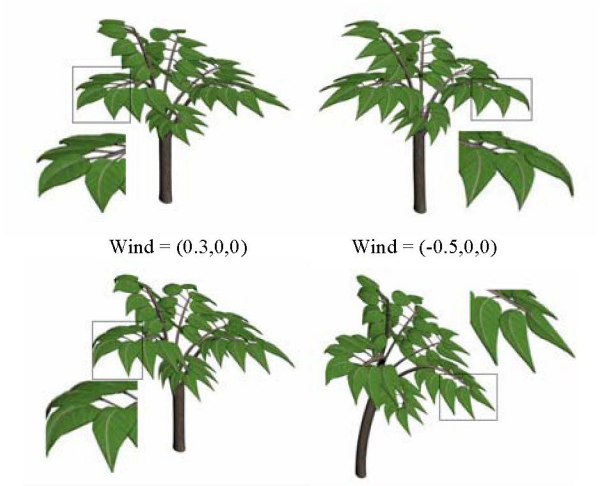
Fig. 6 shows the broad leaf plant model without wind. Fig. 7 shows a major contributor in modifying the value of K_t in the equation (3), and Fig. 8 shows some snapshots of the effects of varying wind vector acting on the plant model when K_t is 0.0002, and shows the views within close distance. At any point, the movement of plant can be dynamically controlled by injecting a global wind into the system by the user. And the structural attributes of a plant like the mass, length, rigidity and the visual attributes like its texture maps also can be interactively specified at run-time by users.

Fig 5. Plant model without external forces

$K_t = 0.002$ $K_t = 0.007$

$K_t = 0.005$ $K_t = 0.015$

Fig 6. Simulation results of adjusting the different values of transfer force parameter K_t.



Wind = (0.3,0,0) Wind = (-0.5,0,0)

Wind = (0.3,0,-0.2) Wind = (0.5,0.2,-0.8)

Fig 7. Simulation results of general broad leaf tree acted by different wind direction.

TABLE I
TIME OF BUILDING AND UPDATING COLLISION MODEL

	All leaves	All stems	A plant
Number of triangle	4752	1764	6516
Time of building OBBs(ms)	104.54	38.80	143.35
Time of updating OBBs(ms)	3.78	3.43	7.21

The broad leaf plant model consists of 3906 vertices and 6516 triangles, including 7 stems and 54 leaves. And each stem has 7 segments and each leaf has 10 cubic tubes as mid-rib or parallel venation segment. The frame rate of the animation was found to run stably at approximately 40-46 fames per second without collision detection, which is realized on the CPU and not on the graphics card, as we do not yet take advantage of any hardware acceleration. Table 1 shows the time of building and updating OBBs in simulating a virtual plant in wind. And it can be seen from it that the refitting speed by using our updating method is faster than re-computing method.

VI. Conclusion

In this paper, we propose a physically based method for realistic rendering and animating broad leaf plant models in real-time with an improved segment method. We creatively apply improved segment based structure to the leaf mode in accordance with the plant's natural attribute. By adding the constraint force to the latitudinal direction, the movements of

the leaves are both rigid and soft. And a new method of collision detection has been proposed, which is based on the combination of spatial decomposition and OBBs. We accelerate the speed of collision detection in two aspects. One of them is to partition the virtual space in homogeneous regions, which reduces the number of OBBs referred to intersection tests. Another is to use of spatially partitioned OBBs, which avoids performing intersection tests with every triangle of the primitive. Experiments show that our method is feasible and suitable to simulate virtual plant in wind. Future investigations will be extended this method to be able to handle vast realistically animated landscape of vegetation and foliage, and the effect of aging and other types of atmospheric variation on the appearance of plants.

ACKNOWLEDGMENT

Many thanks to the entire R&D software team at GRJL for their collaboration during the whole experimentation. This work is financially supported by the NSFC (60573182) and the NSFC (69883004).

REFERENCES

[1] Lindenmayer, A., "Mathmatical models for cellular interaction in development," Parts I and II, Journal of Theoretical Biology, 1968, 18:280-315.

[2] Prusinkiewicz, P., and Lindenmayer, A., The Algorithmic Beauty of Plants. 2004.

[3] Oppenheimer, P. E., "Real time design and animation of fractal plants and trees." ACM Press, 1986, 55-64.

[4] Weber, J. and Penn, J., "Creation and rendering of realistic trees." In Proceedings of SIGGRAPH'95 Los Angeles, ACM Press, 1995, 119-128

[5] Wong, C. J., Datta, A., "Animating Real-time Realistic Movements in Small Plants." ACM press, 2004, 182-189.

[6] Sakaguchi, T., and Ohya, J, "Modeling and animation of botanical trees for interactive virtual environments." In Proceedings of the ACM symposium on Virtual reality software and technology, ACM Press, 1999, 139-146.

[7] Ono H., "Practical experience in the physical animation and destruction of trees," Proc. of Eurographics, 1997, 149-159.

[8] Jakobsen, T., "Advanced Character Physics." In Proceedings of Game Developers Conference, 2001, 383-401.

[9] Gottschalk, S., Lin, M. C., and Manocha, D., OBB Tree: a hierarchical structure for rapid interference detection. In SIGGRAPH 96 Conference Proceedings, New Orleans, USA, ACM, 1996, 171-180.

[10] Provot, X., "Collision and self collision handling in cloth model dedicated to design garments" In Computer Animation and Simulation '97, 1997, 177-189,

[11] Moore, M., and Wilhelms, J., Collision Detection and Response for Computer Animation. Computer Graphics, Vol. 22, No. 4, 289-298.