Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 970617, 8 pages
doi:10.1155/2009/970617

Research Article

Real Time Animation of Trees Based on BBSC in Computer Games

Xuefeng Ao, Zhongke Wu, and Mingquan Zhou

College of Information Science and Technology, Beijing Normal University, Beijing 10087, China

Correspondence should be addressed to Xuefeng Ao, aoxuefeng@mail.bnu.edu.cn

Received 20 January 2009; Accepted 3 April 2009

Recommended by Suiping Zhou

That researchers in the field of computer games usually find it is difficult to simulate the motion of actual 3D model trees lies in the fact that the tree model itself has very complicated structure, and many sophisticated factors need to be considered during the simulation. Though there are some works on simulating 3D tree and its motion, few of them are used in computer games due to the high demand for real-time in computer games. In this paper, an approach of animating trees in computer games based on a novel tree model representation—Ball B-Spline Curves (BBSCs) are proposed. By taking advantage of the good features of the BBSC-based model, physical simulation of the motion of leafless trees with wind blowing becomes easier and more efficient. The method can generate realistic 3D tree animation in real-time, which meets the high requirement for real time in computer games.

1. Introduction

In current computer games, plants in scenes are usually consisted of simple plane pictures positioned in four orthogonal directions. 3D models of plants are seldom used in computer games. Recently, 3D plants come forth in some computer games which make users feel more realistic. For example, in [1], a palm tree model with approximately 400 Polys was created by Amped Labs LLC for the use in the Rise of Power game. There are also some top-level visualization corporations like Interactive Data Visualization, Inc. (IDV) providing functional system like SpeedTree for modeling 3D trees and simulating simple tree animation in computer games [2].

However, in the scope of our knowledge, we do not find any research publications on discussing 3D tree motion in computer games. In fact, many researchers have made contributions in tree modeling and its motion, but none is actually applied in computer games. In tree modeling, the main methods include the followings: L-system [3], image based tree modeling [4–6] and space colonization algorithm [7]. In the aspect of tree motion simulation, there are also many works. The earliest work can be retrieved was done by Wejchert and Haumann [8]. They used four simple fluid flow including uniform, sink, source and vortex to design and control the movements of the wind. And then the animation of the leaves going with the wind is simulated by computing the movement produced by wind force from normal and tangential direction in accordance with the traditional Newton theory. Mikio Shinya created a stochastic wind area and then simulated the tree swaging in the wind based on a modal analysis method [9]. Hiromi simulated tree motions like flying and breaking in tornado in the movie "twister" in which the tornado model was constructed with the turbulence theory [10]. In Feng's study [11], a single branch is divided into several little segments, and each of these segments can be viewed as a pole which cannot deform in the axis direction. Then the position of each point on a little segment after motion can be computed by applying the deformation equations of pole. In Alkagi's work [12], level of detail (LOD) technique was employed to reduce the computational complexity, and the animation of trees in real-time was implemented. During the computation of tree motion, a single branch is divided into seven parts of cone-shaped "links" that are interconnected by six "joints". And the bending of a branch is represented by the rotation of each of its joints. In [13], William Van Haevre realized tree motion at each arbitrary moment using a goal-based motion algorithm. As for recent works, Khalid Saleem animated tree branch breaking and flying effects in a 3D interactive visualization system for hurricanes and storm surge flooding [14]. Yubo Zhang introduced a data-driven approach that synthesizes tree animations from a set of precomputed motion data [15].

However, because of the high demand for real time in computer games, most of the above work cannot be applied directly in computer games. Some can be used to animating trees in computer games like Akagi's work as extra speeding technique was employed to reduce the computational complexity, and thus real-time animation can be generated [12].

There are two main factors hampering the application of 3D tree motions in computer games. For one thing, most of the tree representations are too complicated to implement real-time animation; for another, the simulation of tree animation is a sophisticated work because many physical computations like animation aerodynamics, material mechanics, and pole kinematics are involved.

In our paper, a novel tree model based on BBSC is introduced, and the method of simulating tree motions based on this model is proposed [16]. This model combined with this method is efficient for generating real-time tree motions in computer games. In the following sections, the paper is organized as follows. In Section 2, a novel tree model based on BBSC is described in detail; in Section 3, the model for physical simulation of wind is briefly introduced; in Section 4, the simulation of the tree animation is illustrated; in Section 5, the animation effect by our method is demonstrated, and the conclusion is given.

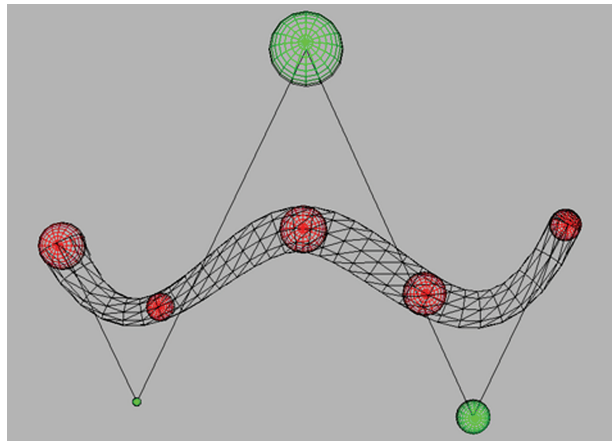
2. BBSC-Based Tree Modeling

Ball B-Spline Curve (BBSC) is a parametric solid representation of freeform tubular objects, which are skeleton-based parametric solid model. BBSC directly defines objects in B-Spline function form by using control sphere instead of control point in B-Spline curve. BBSC not only to describe every point inside 3D solid objects but also provides its center curve in B-Spline form directly. So the representation is more flexible for modeling, manipulation, and deformation.

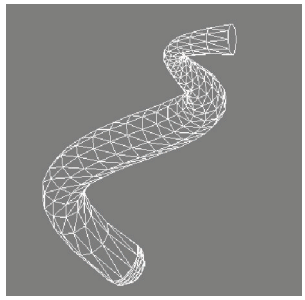
2.1. Ball B-Spline Curve (BBSC). Let $N_{i,p}(t)$ be the ith B-Spline basis of degree p with knot vector $[u_0, \ldots, u_m] = \{a, \ldots, a, u_{p+1}, \ldots, u_{m-p-1}, b, \ldots, b\}$. $\langle P_i; r_i \rangle$ is a ball centered at P_i with radius r_i.

The Ball B-Spline Curve (BBSC) is therefore defined as $\langle B \rangle(t) = \sum_{i=0}^{n} N_{i,p}(t) \langle P_i; r_i \rangle$, where P_i are control points, and r_i are control radii.

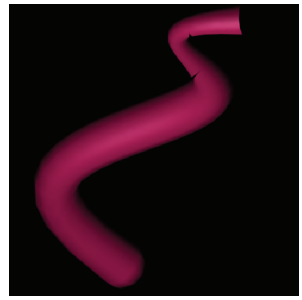
As $\langle B \rangle(t) = \sum_{i=0}^{n} N_{i,p}(t) \langle P_i; r_i \rangle = \langle \sum_{i=0}^{n} N_{i,p}(t) P_i; \sum_{i=0}^{n} N_{i,p}(t) r_i \rangle$, a Ball B-Spline curve can be regard as two parts: a 3D B-Spline curve, that is, the center curve (or skeleton): $c(t) = \sum_{i=0}^{n} N_{i,p}(t) P_i$, and a B-Spline scalar function, that is, the radius function $r(t) = \sum_{i=0}^{n} N_{i,p}(t) r_i$. Therefore most properties and algorithms can be obtained by applying B-Spline curve and function to the two parts of BBSC, respectively. Owing to the perfect symmetry property of balls, the curve $c(t)$ constructed from the centers of balls is exactly the center curve of the 3D region represented by the BBSC. Different from BBSC, B-spline curve only represent a curve in 3D space. But BBSC inherits good properties from B-Spline curves. Most algorithms in B-Splines can be extended to BBSC. For example, we also have interpolation and approximation algorithm generating a BBSC. These algorithms are implemented by employing B-Spline curve's interpolation (approximation) algorithm to position data to get the center curve part of BBSC and B-Spline scalar function interpolation to these widths data to get radius function. Similarly we can modify the 3D shape by deforming BBSCs through modifying its control points and radii. Detailed description of the algorithm can be found in [16]. In Figure 1(a), a BBSC generated by interpolation is shown, in which the red balls are the data spheres, and the green spheres are the control spheres. In fact, the two end spheres are both data spheres and control spheres. The whole curve is tessellated so that later rendering and texture mapping processes can be implemented. In this figure we can easily see that, different from traditional B-Spline curve with 2D control points, the BBSC has the control spheres consisted of center points and radii. Figure 1(b) shows the BBSC in a different viewpoint, and Figure 1(c) is the rendering result of the transformed BBSC.



(a) The data spheres (red) and the control spheres (green)



(b) The BBSC after transformation



(c) The rendered BBSC

Figure 1: A BBSC created by interpolation.

The BBSC presented above has many features which make it very suitable to construct 3D trees in games.

(a) Solid mathematical fundamentals.

(b) Precise evaluation.

(c) Flexibility of manipulations and deformations.

(d) More compact dataset than discrete or linear representations when defining a freeform 3D object.

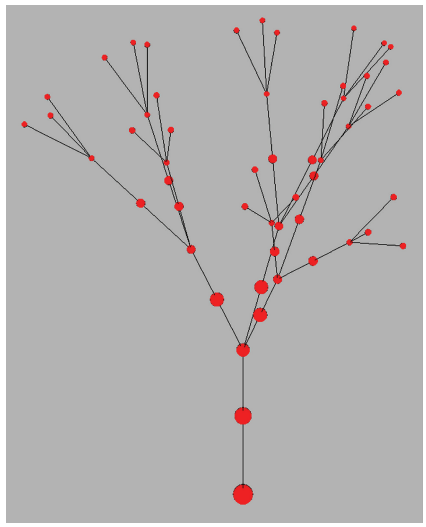


Figure 2: Geometrical model represented by BBSCs.

(a) Mesh mode (b) Rendering result

Figure 3: Tessellated BBSC-based tree.

Therefore, we can generate real-time animation of the BBSC-based trees easily.

2.2. Geometric Representation with BBSC. BBSC is a parametric representation of 3D freeform solid objects [16]. Its evaluation is precise and efficient, and it is flexible for manipulations, deformations, and morphing. These properties provide the potential to build flexible botanical tree model. Figure 2 shows the geometric relationship between these data spheres of a BBSC-based tree. The whole tree is consisted of several BBSCs which are created by interpolation. The red spheres are the data spheres used to be interpolated. Each sphere is consisted of its center point and radius. Each sphere is represented by its center point and radius. Thus a tree is described by these center points and radii of these data spheres. And in Figure 3, the resulting tree constructed from BBSCs is shown. Figure 3(a) is the tessellation result of the BBSC-based tree model, and Figure 3(b) is the rendering result. After tessellation, texture mapping technique can be applied. Therefore, various kinds of trees can be generated through texture mapping techniques in games.

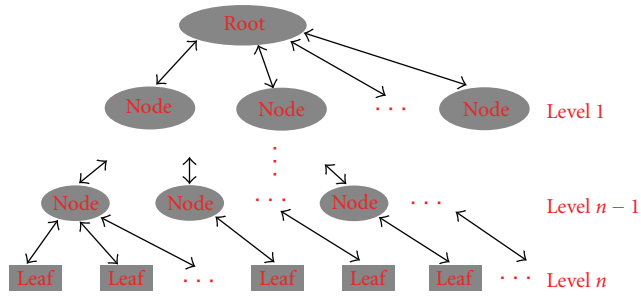


Figure 4: Topological structure of representing a tree.

2.3. Topological Representation. A graph-based data structure (tree data structure) is built to represent the complex hierarchical structures of trees shown in Figure 4. In each node of the tree data structure, the topological information of its parent and children and its geometric representation based on BBSC are stored. The construction of the topological model aims to generate real-feeling animation of the whole tree. The hierarchical structure will be made use of to compute the movements of the branches from low level to top level.

3. Model for Physical Simulation of Wind

In the wind model, we adopt Feng Jinhui's method for physical simulation of wind [11, 17]. Here, a summary of the method is given.

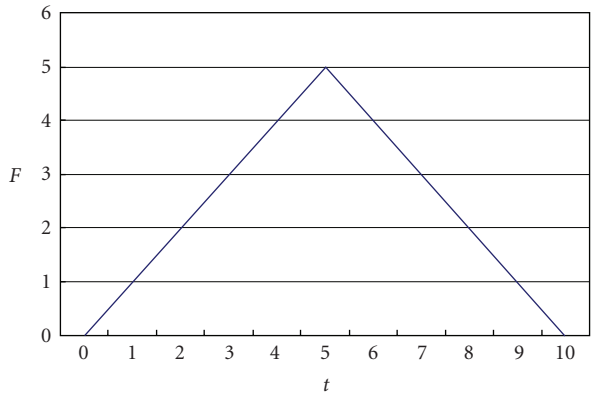
3.1. Wind Force Size. In our method, the users are allowed to choose the wind force model and set the wind direction according to their requirements. Two kinds of wind force model are provided, and arbitrary wind direction in x-z panel can be set.

3.1.1. Gust of Wind. The gust of wind increases gradually from zero to the highest point and then decreases gradually to zero again. The model can be represented in the following equation:

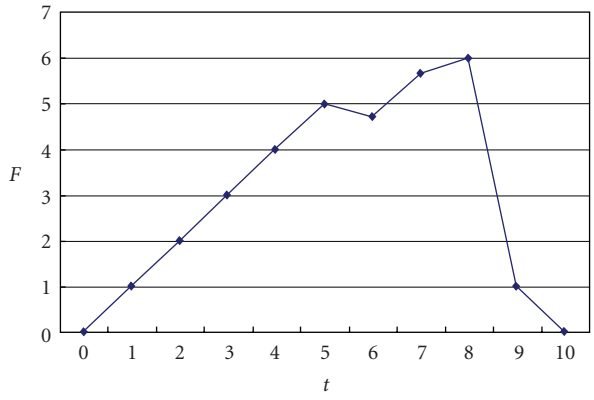
$$F_1 = \begin{cases} at + b, & 0 \le t \le t_c, \\ \dfrac{c - d(t_c - t)}{t_c}, & t_c \le t \le \text{max\_time}. \end{cases} \quad (1)$$

3.1.2. Stable Wind. The stable wind increases gradually from zero to certain grade, and for some seconds retains at this grade, then finally decreases gradually to zero. The model can be represented in the following equation:

$$F_2 = \begin{cases} at + b, & 0 \le t \le t_c, \\ c + d\sin(t), & t_c \le t \le \text{mid\_time}, \\ \dfrac{e - f(t_c - t)}{t_c}, & \text{mid\_time} \le t \le \text{max\_time}. \end{cases} \quad (2)$$



(a) The gust of wind model



(b) The stable wind model

Figure 5: The relationship between the wind force and the time.

In the above two equations, t_c is the time constant, and different wind models can be easily obtained by modifying the model parameters.

Figure 5 shows the wind force changes with the time. Figure 5(a) gives an example of the gust of wind changing with the time, and Figure 5(b) is the example of the stable wind changing with the time.

3.2. Wind Force Direction. Users are allowed to set arbitrary wind direction in x-z panel by inputting the angle between the wind direction and the x-axis positive direction. 360 wind directions along with the counter-clockwise can be obtained by increasing the angle from zero degree to 360 degree.

The later computation of the motion of the branches is based on the above wind model.

4. Animation of BBSC-Based Trees

In the introduction part, we have noticed that when computing the motion of a certain branch, those researchers generally segment a certain tree branch into several segments and then view those little segments as poles. Therefore, the deformation method of a pole can be applied to the little segment very easily to generate relatively natural-looking tree animation.

As described in Section 2, our tree model based on BBSC is a proper and efficient model in computer games. In fact, this model shows more value when computing tree motion. Now that the branches are generated from several data spheres within, we can just use the position of the data spheres to segment the current branch. And considering a branch is an actually BBSC created by interpolating several data spheres, we need only compute the position of the data spheres after motion rather than every point within the segment. The new curve obtained by interpolating the new data spheres after motion is hence regarded to be the new representation of the branch after motion. With this model, the deformation of tree branches can be computed by defining the relationship between wind forces and data spheres repositioning. And the new position of the data spheres can be computed by simulating the bending of a pole.

4.1. Dynamics Model for Branches. As shown in Figure 6(a), a pole with one end A fixed bends under the distribution force q. Then the displacement and the rotation angle can be computed for each position x. According to pole kinematics theory, the deformation of a pole can be represented by two parameters: the deflection and the rotation angle. The deflection can be just viewed as the displacement of the current position. However, in order to lessen the computation complexity, we consider only the rotation angle in our model. Furthermore, rather than each point in the current segment, we should only consider the rotation angle of the end B under force q, which can be obtained from the following equation:

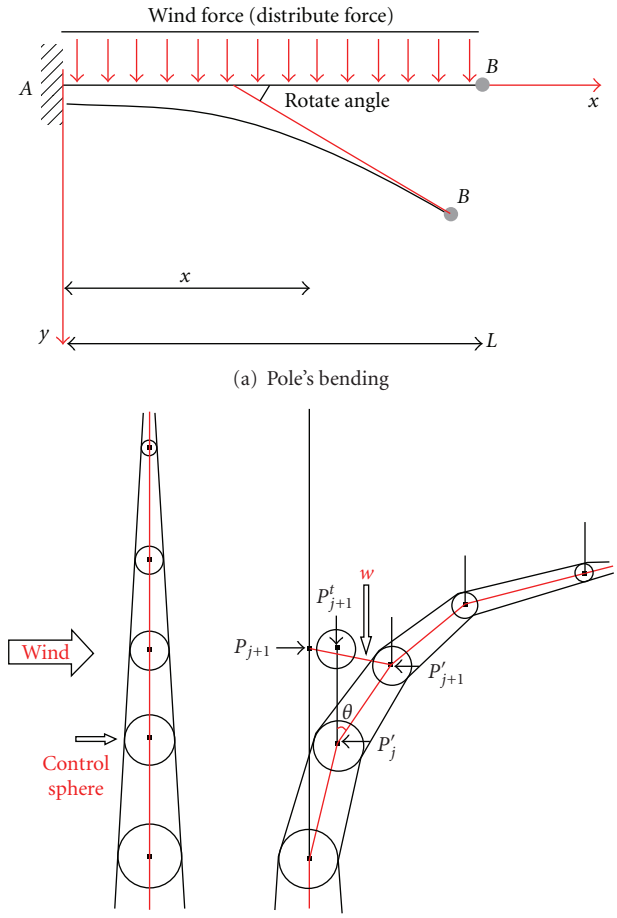
$$\theta_B = \frac{qL^3}{6EI_Z}. \tag{3}$$

In the above formula, q is the wind force, L is the length of the pole, and I_Z is the Bending Section Modulus. E is the Young's Modulus, which is used to measure the elastic characteristic of certain materials and is decided only by the physical feature of the material. We indeed have omitted many complex computation processes which are indispensable in the field of Mechanics of Materials; however it is fully accepted in computer games for realistic.

And the BBSC-based tree's bending by simulating a pole's bending is shown in Figure 6(b), in which P_{j+1} is the original coordinates of the current moving point, P'_{j+1} is the coordinates of the points after moving, P^t_{j+1} is the position of P_{j+1} after horizontal translation, and P'_j is the coordinates of the former points after moving within the current branch. The rotation angle under the wind from x-axis direction is θ. By applying (3), we get

$$\theta = \frac{F_{\text{Wind}} \cdot \text{dist}\left(P'_j, P^t_{j+1}\right)}{6EI_Z}. \tag{4}$$

In the above equation, F_{Wind} is the wind force, and $\text{dist}(P'_j, P^t_{j+1})$ is the distance between P'_j and P^t_{j+1} which represents the length of the current segment.



(a) Pole's bending

(b) BBSC model's bending

Figure 6: BBSC model's bending by simulating pole's bending.

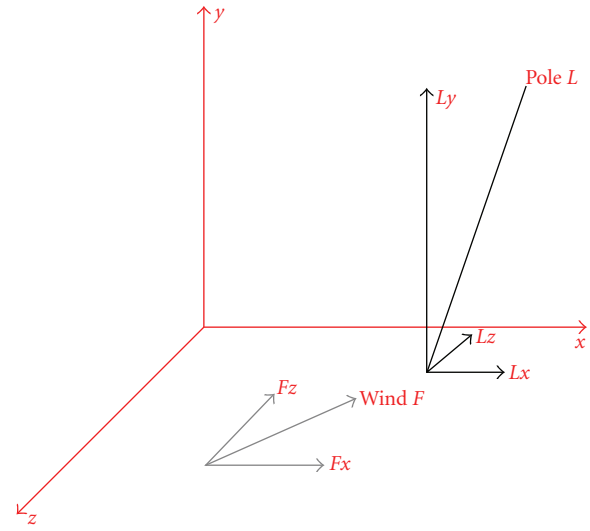


Figure 7: Wind decomposition and pole decomposition.

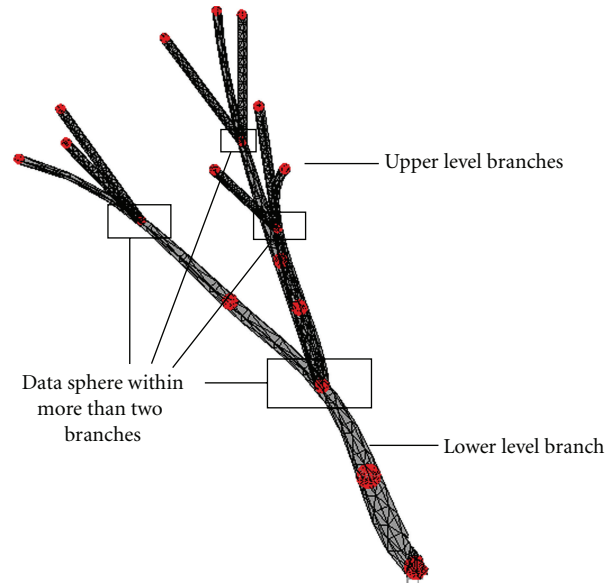


Figure 8: Data spheres contained by more than two branches.

4.2. Solution of Motion. In our model, the branches have been already divided into several segments, and each segment between two data spheres can be viewed as a pole. Considering the high request for real-time in computer games, we just apply simple deformation method of a pole to each segment. Therefore only bending is taken into account. The rotation along the cross section and the deformation along the axis direction are both neglected. The bending of a branch can then be described as three Eula angles which represent the rotation angles of the data spheres around the x-axis, y-axis, and z-axis, respectively.

Suppose the coordinate vectors of the data point before and after deformation are (x, y, z) and (x', y', z'), respectively. Then the two vectors has the relationship as in the following equation:

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = [R] \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}. \tag{5}$$

In the above equation, R is the rotation matrix, which can be described as in (6). The rotation sequence is as follows: firstly, rotates around z-axis by angle θ_z; then rotates around the rotated y-axis by angle θ_y; finally rotates around the rotated x-axis by angle θ_x:

$$[R] = \begin{bmatrix} c\theta_y c\theta_z & c\theta_y s\theta_z & -s\theta_y \\ s\theta_x s\theta_y c\theta_z - c\theta_x s\theta_z & s\theta_x s\theta_y s\theta_z + c\theta_x c\theta_z & c\theta_y s\theta_x \\ c\theta_x s\theta_y c\theta_z + s\theta_x s\theta_z & c\theta_x s\theta_y s\theta_z - s\theta_x c\theta_z & c\theta_y c\theta_x \end{bmatrix}. \tag{6}$$

In the above equation, c and s are the cosine and sine values of the related angles. And the position of any data points after deformation can be computed with this rotation matrix as long as all the angles have been figured out.

The three angles can be obtained by applying the pole deformation theory under the situation of decomposing the wind vector and the pole segment vector, respectively. The wind force vector can be decompounded into two vectors

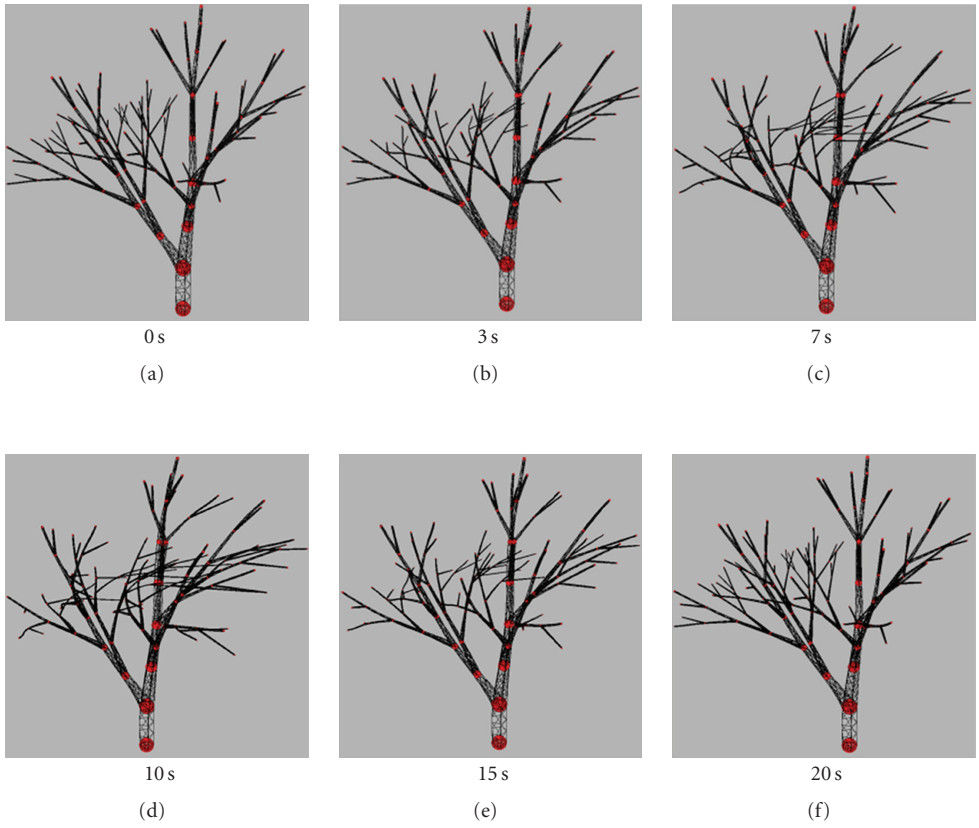


Figure 9: Six states extracted from the 20 seconds animation of a tree.

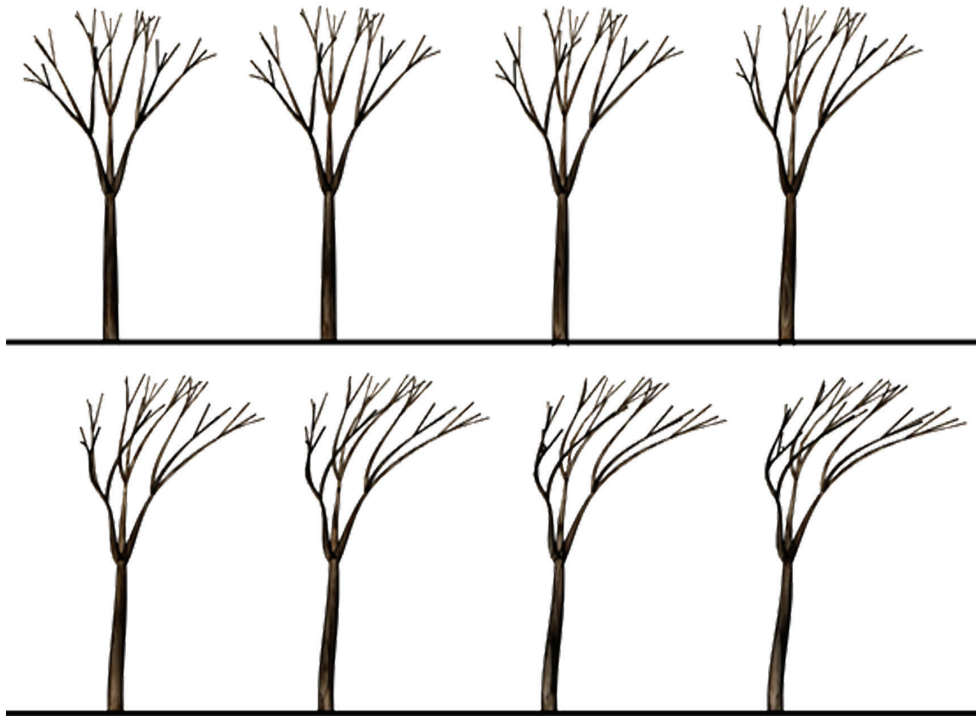


Figure 10: Eight states extracted from the 40 seconds animation of a tree.

which are along x-axis and z-axis as the wind force always lies in the x-z panel. And the vector between the start point and the end point of a segment can be decompounded into three vectors which are along x-axis, y-axis, and z-axis, respectively. The decomposition process is illustrated as in Figure 7.

Then there are four situations about the wind force acting on the segment vector.

(a) The wind force along the x-axis acting on the segment vector along the y-axis leads the segment to rotate around the z-axis by angle α_z.

(b) The wind force along the x-axis acting on the segment vector along the z-axis leads the segment to rotate around the y-axis by angle α_{y1}.

(c) The wind force along the z-axis acting on the segment vector along the x-axis leads the segment to rotate around the y-axis by angle α_{y2}.

(d) The wind force along the z-axis acting on the segment vector along the y-axis leads the segment to rotate around the x-axis by angle α_x.

In each above situation, the related angle can be computed through (4).

And finally, the rotation angle of the data points can be computed as follows:

$$\begin{aligned} \theta_x &= \alpha_x, \\ \theta_y &= \alpha_{y1} + \alpha_{y2}, \\ \theta_z &= \alpha_z. \end{aligned} \tag{7}$$

The rotation matrtix for the current data sphere can be obtained through (6). Then, the position of the data sphere after motion can be figured out by multiplying the rotation matrix to the original vector of the data sphere as (5).

4.3. Movements of the Whole Branch. In fact, the solution of motion described as above just aims to the little segment between two data spheres. And the motion of the whole branch is obtained by computing the motion of its data spheres from bottom to top one by one and then interpolating the new data spheres after motion. For a backbone branch, the initial data sphere is the root; otherwise the initial data sphere is also within another branch. The root data sphere is obviously not moving. But the motion of the data spheres contained by more than two branches should be computed carefully. If the current branch is an upper level one, then the position of the initial data sphere could just employ the position obtained in lower level branch. For example, in Figure 8, the data sphere bounded by the rectangle is contained by three branches. Then for the upper level branches, the motion of the initial data sphere doesn't need to be computed anymore as it can be obtained by employing the motion which has been computed in the lower level branch directly.

4.4. Animation of the Whole Tree. As the tree has been constructed in accordance with the hierarchical structure as in Figure 4, the motion of the whole tree can be obtained by computing the motion of the branches from root branch to the leave branch hierarchically. The depth-first traverse algorithm is employed to solve the computing sequence problem.

5. Results and Conclusions

5.1. Results. Giving related parameters and certain time t, the movements of each data sphere, furthermore each branch, and finally the whole tree under the wind force in the current moment can be gotten. And by giving a period of time, we can get the continuous animation of the tree under different wind model. In Figure 9, the states of one tree in the moment of 0 seconds, 3 seconds, 7 seconds, 10 seconds, 15 seconds, and 20 seconds during 20 seconds animation under the gust of wind from x-axis direction are shown. In Figure 10, eight states of one tree during the 40 seconds animation under the gust of wind from x-axis direction are demonstrated.

5.2. Conclusions. In this paper, an approach of generating real-time animation of trees based on a novel model—BBSC—is proposed, which can be applied in computer games. BBSC is a good representation for 3D trees and plants in particular for computer games as its solid mathematical representation and more compact dataset. Moreover, animation of trees can be generated easily as the data spheres have divided the branches into little segments automatically thus the motion of the tree can be obtained by computing the motion of each of these data spheres. By interpolating these data spheres after moving, the motion of the branch, and finally the motion of the whole tree can be implemented. The experimental results show that this method is proper and efficient for simulating tree animation in computer games.

Acknowledgment

The project is sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

References

[1] M. Mitman, "Free Palm Tree," GarageGames, http://www.garagegames.com/community.

[2] "SpeedTree," Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/SpeedTree.

[3] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer, New York, NY, USA, 1990.

[4] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," *ACM Transactions on Graphics*, vol. 26, no. 3, article 87, pp. 1–7, 2007.

[5] C.-H. Teng, Y.-S. Chen, and W.-H. Hsu, "Constructing a 3D trunk model from two images," *Graphical Models*, vol. 69, no. 1, pp. 33–56, 2007.

[6] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," *ACM Transactions on Graphics*, vol. 26, no. 3, article 88, 2007.

[7] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Proceedings of the*

Eurographics Workshop on Natural Phenomena (EGWNP '07), pp. 63–70, Prague, Czech Republic, September 2007.

[8] J. Wejchert and D. Haumann, "Animation aerodynamics," in *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*, pp. 19–22, New York, NY, USA, July 1991.

[9] M. Shinya and A. Fournire, "Stochastic motion—motion under the influence of wind," in *Proceedings of the Computer Graphics Forum (Eurographics '92)*, pp. 119–128, Cambridge, UK, September 1992.

[10] H. Ono, "Practical experience in the physical animation and destruction of trees," in *Proceedings of the 8th Eurographics Workshop on Computer Animation and Simulation*, D. Thalmann and M. van de Panne, Eds., pp. 149–159, Springer, Budapest, Hungary, September 1997.

[11] J. H. Feng, Y. Y. Chen, T. Yan, and E. H. Wu, "Going with wind—physically based animation of trees," *Chinese Journal of Computers*, vol. 21, no. 9, pp. 669–773, 1998.

[12] Y. Akagi and K. Kitajima, "Computer animation of swaying trees based on physical simulation," *Computers & Graphics*, vol. 30, no. 4, pp. 529–539, 2006.

[13] W. V. Haevre, F. D. Fiore, and F. V. Reeth, "Physically-based driven tree animations," in *Proceedings of the Eurographics Workshop on Natural Phenomena (EGWNP '06)*, pp. 1–8, Vienna, Austria, September 2006.

[14] K. Saleem, S.-C. Chen, and K. Zhang, "Animating tree branch breaking and flying effects for a 3D interactive visualization system for hurricanes and storm surge flooding," in *Proceedings of the 9th IEEE International Symposium on Multimedia Workshops (ISMW '07)*, pp. 335–340, Taichung, Taiwan, December 2007.

[15] L. Zhang, Y. Zhang, Z. Jiang, L. Li, W. Chen, and Q. Peng, "Precomputing data-driven tree animation," *Computer Animation and Virtual Worlds*, vol. 18, no. 4-5, pp. 371–382, 2007.

[16] H. S. Seah and Z. K. Wu, "Ball B-Spline based geometric models in distributed virtual environments," in *Proceedings of the Workshop towards Semantic Virtual Environments (SVE '05)*, pp. 1–8, Villars, Switzerland, March 2005.

[17] J. H. Feng, *Going with Wind—Physically-Based Animation*, Institute of Software, Chinese Academy of Science, Beijing, China, 1999.