An Implementation on Real-time Animation of Trees Swaying in Wind Fields

Cansin Yildiz,
Bilkent University, Computer Engineering Department
06800 Ankara, Turkey
cansin@cs.bilkent.edu.tr

Abstract. Trees are one of the most important objects of everyday life. Therefore, it is really crucial to have a realistic tree animation, especially for virtual reality applications. Many studies have been made on simulating tree motion under influence of wind. Most of the time, those studies use physical based approaches to generate the branch motions. In this paper, I propose a hybrid method to create natural motion of branches using both physical equations and stochastic effects, based on [1]. This method enables real-time animation of branches.

Keywords: tree, motion, wind, spring-force model, real-time animation.

1 Introduction

An accurate representation for vegetation is one of the most important topics in computer graphics field. Starting with late 80’s, a lot of research has been done concerning this issue. However, most of those studies [2] [3] [4] are solely focused on modeling the shapes of trees and vegetation, while ignoring motion. But, fortunately, there are also works which incorporate model with motion [1] [5] [6]. Most of these methods generate branch motion using pure physical based approaches. Although, physical approaches can create very accurate motion of branches under influence of wind and other external forces, their calculation complexity makes real-time execution hard. There is also a completely different approach for plant motion problem; to use motion capture data as guidance for generating new plant motions [10].

To the author’s knowledge, there are few methods which can realize real-time execution [1] [7] [8] [9]. Similar to what is introduced by Ota et al. [1], this paper proposes an efficient way of creating natural branch motion. The method is a hybrid one, which combines spring-force model for general motion and stochastic sine oscillation for randomness. This method can easily handle different kinds of trees, like weeping trees (see Fig. 1).
In section 2, I give a brief description of overall process. Then, I continue with giving the details of simulation step of our method. After giving the details of rendering step at section 4 as well, I introduce the results that are obtained from this work at section 5. I conclude the paper with some discussion on future work and so forth at section 6 and 7.

2 Overview of the Method

As all animation methods, the approach has two main concerns; simulation and rendering. For simulating trees’ branch motion, a physical based approach is used. The general motion of each branch is simulated as a mass-spring system. Using the displacement caused by the external wind force, a displacement angle is calculated for each branch segment, and it is accumulated through their children as well. The resulting simulation is further enriched by adding oscillation behavior of sine-based stochastic force. For that simulation to be as realistic as possible, some other measures are also taken, which is discussed deeply in following section.

The other issue that has to be handled was rendering. Basically, the concern is to have a tree rendering as realistic as possible. To do that some problems are needed to be overcome like skinning the tree, solving breaking of branch segments, etc. Another issue, which is by no ways less important, is to have a supportive environment for tree animation. The details of both rendering concerns are investigated at section 4.

3 Simulation

The main concern of proposed method in this paper is to simulate branch motions of trees. For the sake of simplicity, the leaf motion is ignored. To represent the wind effect on branches, I adapted a simulation method based on cantilever spring model as proposed at [1]. The overall simulation that is obtained from this cantilever spring model is supported by an oscillation simulation using sine-based force functions. The
rest of this section is first explains the branch model that is used to represent the tree, and how it is actually simulated.

### 3.1 Branch Model of a Tree

The model for representing a tree is consists of branches. Between each two consecutive joints, there is at least one branch segment (see Fig. 2), but there can be multiple segments as well to have a soft motion of long branches; which is most common in weeping trees. Each branch segment is given some properties, namely, length, base width and initial angles x/y. Top width of a branch segment is calculated according to its children’s base widths.

![Fig. 2. Branch segments and their properties.](image)

### 3.2 Simulation Based on Spring Model

Each branch segment has its own local coordinate system, which makes simulation easier. The origin of this local coordinate system is set to bottom of the branch segment, and axes are perpendicular to branch direction (see Fig. 3).

![Fig. 3. Local coordinate system of a branch segment.](image)

The simulation method is based on cantilever flat-spring model, as proposed by Ota et al. 1. As they described, a branch segment is approximated as a squared non-elastic wood, and the displacement amount is calculated from the wind load given to
the branch segment. The pressure that is applied to the branch along x/y directions is described as;

\[
\begin{align*}
P_x(t) &= F_x(t) \times (1 + a \ast \sin(t + b)) \\
P_y(t) &= F_y(t) \times (1 + a \ast \sin(t + b))
\end{align*}
\]

Here, the first term \( F(t) \) is the directional force that can be applied to a branch segment. For current purpose, it is used as the wind force. Although the coefficient that is applied to \( F(t) \) may seem complicated, it is pretty straight-forward. \( l \) is to represent the wind force’s itself. \( a \ast \sin(t+b) \) coefficient is the so called oscillation behavior of the branch segment. By having a coefficient \( a \) in front of sine function, the current strength of oscillation is controlled. \( a \) is a small number, which is derived from the current angle of the branch segment. Basically, if a branch segment is bended much, it will oscillate more as well. The randomness of this oscillation is realized by \( b \) constant. It is basically a random number, which makes each branch segment to be in different sine phases at same time step.

Using those loads, displacement amounts of the branches are calculated as;

\[
\begin{align*}
\delta_x(t) &= P_x(t)/k \\
\delta_y(t) &= P_y(t)/k
\end{align*}
\]

Here, \( k \) is the spring constant of branch segment. It is determined by;

\[
k = \frac{Ebt^4}{4l^3}
\]

\( E \), the elastic modulus, is specific to each tree. Whereas, the width \( b \), thickness \( t \) (base width - top width), and length \( l \) is specific to each branch segment.

After calculating the displacement amounts using spring constant and load, the actual motion angles \( \Theta(t) \) are needed to be calculated (see Fig. 4). Those angles are calculated as;

\[
\begin{align*}
\Theta_x(t) &= \sin^{-1}(\delta_x(t)/l) \\
\Theta_y(t) &= \sin^{-1}(\delta_y(t)/l)
\end{align*}
\]
It must be noted that, the final motion angles are further restricted to not exceed a certain degree $ \alpha $ (20° in current implementation) to force the natural behavior of tree branches.

Although this method uses a physical approach, it can maintain real-time performance because of the flat-spring model’s simplicity.

### 3.3 Integration of Branch Motions

The motions of individual branch segments are obtained as described at 3.2. After that they must be integrated to form the final motion of the actual tree. As shown at Fig. 5, the branch segments’ motion is summed up from parent to children branches.

![Fig. 5. The accumulation of motion through parent to children segments.](image)

### 4 Rendering

A successful simulation should always be supported with a realistic rendering. There are two main concerns about rendering; rendering a realistic tree and a beautiful scene that the tree can live in. The rest of this section gives the details of those rendering issues.

#### 4.1 Tree

Unlike most computer animation problems, rendering an acceptable tree is a straightforward problem. To achieve that, the tree branches are rendered as cones and covered with a decent bark texture. The only problem about this approach was that there could be some breaking conditions; i.e. when the angle between two consecutive branch segments are larger than a value, it could be seem like the branch is breaking. This notion is overcome by simply rendering spheres of proper radius between each branch segments (see Fig. 6).
Fig. 6. Breaking problem is handled using spheres of appropriate radius.

4.2 Scene

Plate that used is always as important as the food served. Therefore, a realistic scene for tree animation is a top priority. Using only simple OpenGL tricks, a decent scene is created. This scene has some fine details like a sky-dome with a nice texture, a skyline between sky and ground, and even the little grasses that are near root of tree trunk. Those details can be seen at Fig. 7.
5 Results

In this section, the results that are obtained from the actual implementation will be demonstrated. But, since it is a tree animation, I urge you to watch the supplementary videos.

The first figure (Fig. 8) is a regular tree which sways under effect of wind. You can observe how overall movement is consistent with general wind direction yet has randomness as well.

![Fig. 8. A regular tree under effect of wind. White arrows represent wind direction and magnitude.](image)

The second result is more interesting. The system is tested for a different kind of tree; a weeping tree (see Fig. 9).

![Fig. 9. A weeping tree under effect of wind.](image)

In weeping tree test, it is realized that an important implementation detail was skipped. When the branch segments lays upside-down, the wind force should decrease their angle, rather than increasing it (see Fig. 10). After adding that little specification, system is fully suitable for all kinds of trees.
Last tree that is simulated is a trunk of palm tree (see Fig. 11). This run nicely represent the notion that a trunk can consist of several branch segments, to simulate the flexible behavior.

6 Future Work

This system is currently flexible enough to represent most kinds of branch motion. An obvious extension should be the simulation of leaf motion as well. Although branch motion has two degree of freedom (twisting rotations are ignorable), it is not the case for leaf motion. So, different simulation logic should be implemented for leaf motion.

Another interesting extension for the current work could be to implement a tree generation algorithm as well with a growth simulation. So far, tree models that have shown are generated by manual means, which is a long tedious work.
7 Conclusion

Final state of the implementation is really satisfactory, though; there is always room for cleaning and clarifying the code, as all application has. Without adding any new features to the system, only improvement that comes to mind is to port the simulation logic from CPU to GPU, for better performance. Other than that, it is a decent simulation of trees swaying in wind.

References