Animating Real-time Realistic Movements in Small Plants

Jason C. Wong*
The University of Western Australia

Amitava Datta†
The University of Western Australia

Abstract

Much of the research involved in computer graphics is focused on creating realistic images and animations that mimic the world we see around us, as well as creating believable environments not from this world. Techniques for animating realistic water, smoke, fire, fog, and other natural phenomena have been extensively explored. It is only recently that powerful computer hardware has become available to achieve these realistic animations.

Compared with other natural phenomena, animations of vegetation and foliage are relatively undeveloped. There are many instances where animations of vegetation and foliage are used in non real-time applications such as special effects in movies. Foliage in real-time applications on the other hand have usually been limited to simple textures and other techniques that do not represent realistic movements at all.

We propose a method for rendering and animating small plants in real-time and in a realistic manner. These real-time animations will be suitable for many real-time applications such as games and realistic interactive virtual environments.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

Keywords: real-time, animations, modeling, small plants, foliage

1 Introduction

Throughout the history of computer graphics, a recurring goal of scientists and artists alike is to reach the point where graphics and reality become indistinguishable. For artists and game programmers, the realism of graphics can often determine the immersiveness of a program. For scientists, the realism is needed to ensure the accuracy of simulations and predictions of object behaviour. There are many other reasons for striving for realistic graphics and the area of plant and vegetation modeling is no different.

There has been a substantial amount of work on modeling plants. There are methods that focus on the realism of the plant models and some that concentrate on the animations of such plant models. Prusinkiewicz and others have developed sophisticated algorithms to generate botanically accurate plant models [Prusinkiewicz et al. 1988; Prusinkiewicz et al. 1993; Prusinkiewicz and Mndermann 2001; Noser et al. 2001; Power et al. 1999; Weber and Penn 1995]. On the other hand Deussen et al. and Sakaguchi and Ohya developed techniques to animate complex landscapes and interactive environments [Deussen et al. 2002; Sakaguchi and Ohya 1999].

These generated plant models are usually computationally expensive to animate. To counteract this, optimizations (such as "level of detail" or LOD in the work by Deussen et al. [2002]) are needed to bring animations into framerates acceptable to be classified as real-time. Because of the complexity of the models and the intricacy in moving them, realistic animations of foliage are largely restricted to non real-time applications. These animations of foliage are quite widely used in special effects in modern feature films. Examples include the "Lord of the Rings" trilogy where the creation of the Ent characters is aided by computer generated and animated foliage [Aitken and Preston 2003].

There have been attempts in creating interactive virtual environments such as works by Guerraz et al. [2003] and Sakaguchi and Ohya [1999]. Guerraz et al. concentrates mainly on using LOD in rendering and animating a field of grass using a texel based modeling technique. Their work involves representing blades of grass in close proximity to the camera by billboarding techniques. Grass that is further away are represented by other texture based techniques. This method produces good results for large fields of grass, but it is not very well suited for extreme close ups.

Sakaguchi's work on the other hand involves gathering 3D volumetric data on real trees to create a realistic model. They then discuss their approach in animating the tree model. They propose a method of segmenting the branches and calculating the rotation and translation caused by external forces.

The algorithms used by Sakaguchi to calculate the branch movements are based on complex physics such as moments of inertia, axial damping, and back propagated forces. These consequently contribute to the intensive computations needed. The tree models used were also simplified by using simple polygons to represented leaves. They assume that there is no need to simulate movements of individual leaves for large tree models.

Since there is currently no efficient method to render small plants where the leaves are also significantly affected by external forces, we propose a method to render and animate small soft plants. Besides the branches, we also segment the leaves, which allows the leaves to move realistically as well. We also propose a versatile algorithm that would be suitable to model the movements of a large range of plant models from broad leaf plants to long thin wheat or grass.

2 Our Approach

In calculating the rotation and translation of the stems and leaves, we propose a simple algorithm that takes into account physically based forces to determine the overall position of the whole plant. In particular, we recognise and implement the forces of gravity and wind which are the main factors in determining the resulting shape and behaviour of the plant.

We do not model the true physics of the movements, but rather opt for a method that approximates the plant movements. We intend for this animation method to be extended to a large number of plant models in order to create a whole landscape of vegetation. We also

*e-mail: jasonw@csse.uwa.edu.au
†e-mail: datta@csse.uwa.edu.au

intend for this method to be suitable for real-time applications and still provide a visually realistic representation of vegetation.

2.1 Model Segments

Our approach is based on the concept of representing a leaf or stem model by a series of thin strip segments. Each of these segments is connected to its adjacent segments and are of equal length. An example of the segmentation of the models is shown in Figure 1. These segments act similarly to the particle systems used in modeling cloth [Choi and Ko 2002] and in modeling snakes and worms [Miller 1988].

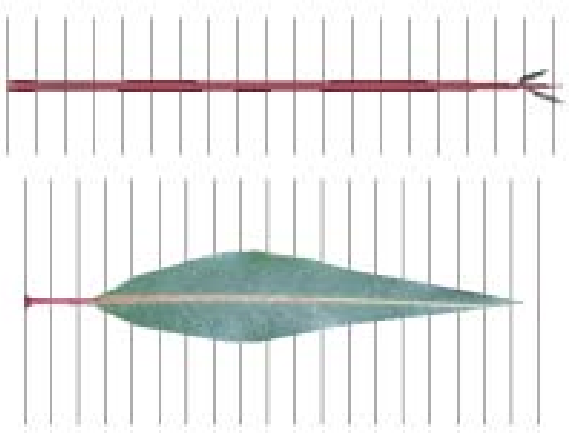

Figure 1: A sample segmentation of the stem and leaf models. Note that the segmentation is dynamically generated and is easily configurable.

During segmentation the initial models of both the stem and leaves remain unchanged, but each vertex in the models is tagged to a particular segment. When the segment is moved, the vertex also moves. In this way, the connected segments act as an animation skeleton for the model.

Each of the models of stems or leaves is segmented during runtime. This dynamic generation of the segments provides the option to easily alter the number of segments which will determine the detail of possible movement. The more segments that are used to represent a model the finer the movements that can be achieved.

2.2 Rigidity

To provide a robust and flexible technique to animate realistic plant movements, it is necessary to include a variable that can ultimately determine the behaviour of the plant. We implement this feature through the notion of *rigidity*.

We model plants with two major components, namely the stem of the plant and the leaves. Both the stem and the leaves have rigidity parameters. Since stems are more rigid than leaves, it is intuitive that the stem rigidity is higher than the leaf rigidity. This simply means that it requires more force to affect the stem than it does for the leaves.

Each segment has its own rigidity value. This allows the rigidity to vary throughout a leaf or stem. It is also intuitive that the rigidity of both leaves and stems is decreased along the length similar to the concept of cantilevers [Weisstein 2003]. The further the distance of a segment from the branch or root, the lower the rigidity.

It is the rigidity parameter that determines the effectiveness of an external force acting on the leaf/stem. In the case of the leaf, the rigidity term is used in determining how much roll, pitch, or yaw rotations are applied to the leaf (see Section 3.1).

The rigidity of a leaf or stem is composed of three values corresponding to the rigidity in the x, y, and z directions. Each of these values determine the effective proportion of the force acting on the leaf or stem. For example, a stem with a high rigidity in the x direction and low rigidity in the y direction will resist a wind force blowing in the x direction more than airflow blowing in the y direction.

Therefore, by varying the value of the rigidity, it is possible to control the general behaviour of the plant. Hence, rigidity makes it possible to model different types of plants. Plants such as shrubs will have a high rigidity since they are rigid and strong, whilst smaller plants like sprouts and long thin plants like wheat will have a very low rigidity, lending themselves to be significantly affected by the wind and gravity.

2.3 Leaf Shape Due to Gravity

Given that rigidity varies throughout the leaf, the force of gravity has interesting effects on the leaf's shape. We make the assumption that the gravity force on each of the segments remains constant since the segments are roughly the same size.

Since the rigidity already decreases along the length of the leaf, there is no need for increasing the gravity force along the leaf that usually occurs with the accumulative weight of the segments. In this sense, the rigidity also takes into account the increasing weight along the leaf, causing the leaf to 'droop'.

Intuitively, the rigidity within the leaf is highest when the segments are close to the stem, and lowest when the segments are furtherest away from the stem. The decrease in the rigidity occurs due to the slowly accumulating weight of the leaf. That is, the further away from the stem, the more weight it has to hold up. Hence, only a small amount of force is needed to move segments that are far from the stem.

With the decreasing rigidity of the leaf, the rotation due to gravity will increase with each segment. Therefore, with a constant gravity force, the shape of the leaf can be solely determined by the rigidity. In other words, the function of change in the rigidity will ultimately determine the shape of the leaf. This is extremely useful for modeling the behaviour of different species of plant leaves by simply changing the rigidity. An example of the leaf shape being defined by the rigidity and gravity is shown in Figure 2.

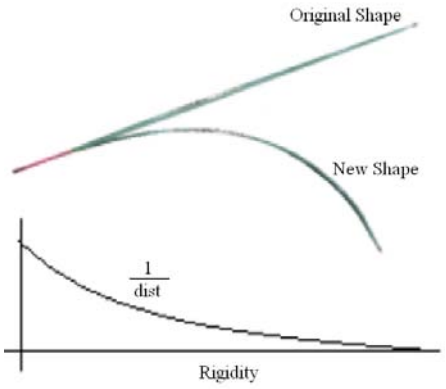

Figure 2: The effect of constant gravity over the segments with varying rigidity. The original leaf shape also shows the original orientation of the leaf. The new leaf shape shows the effect of applying gravity. The rigidity follows the function of $\frac{1}{distance}$.

3 Rotating Segments

We found that movements of the leaf and stem can be suitably achieved through a series of rotations on the model segments. There is no displacement explicitly defined since the rotations in the connected segments will produce the necessary displacement. Therefore, through inspiration from the movement management used by Lowe et al. in animating a 3D human avatar [Lowe et al. 2002], the segments were manipulated purely through rotations. An example of how rotating the segments can cause overall displacement is shown in Figure 3.

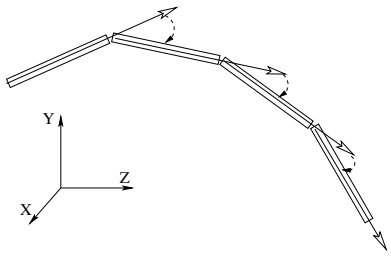

Figure 3: The effect of progressively rotating each of the segments leading to overall displacement. It is through this concept that the leaf and stem movements are modeled.

We observed that leaves in nature are governed by rotations in the x, y, and z directions which translate into the yaw, pitch, and roll rotations on the leaves respectively. The rotations of roll, pitch, and yaw are shown in Figure 4.

Quaternions were used to represent these rotations to prevent Gimbal lock that can occur in using Euler Angles [Ramamoorthi and Barr 1997]. Quaternions are considered an extension of complex numbers and can be used to represent rotations in three dimensional space [Weisstein 1999]. Quaternions contain a three dimensional rotational axis and a rotational angle. Note that the rotation axis used in these quaternions do not correspond to the directions. For example, pitch is a rotation in the y direction, but it is achieved through a rotational axis in the x direction.

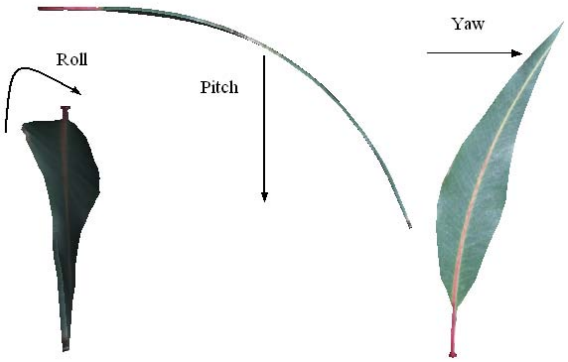

Figure 4: The possible rotations in a leaf. The yaw of the leaf is in the x direction. The pitch of the leaf is in the y direction. The roll of the leaf along the z direction.

The calculation of rotations and the application of these rotations will be discussed in detail only in the case of leaf segments, as the process is almost identical in the case of rotating stem segments. However, roll rotations do not easily occur on the stem, so the calculation of roll rotations is not necessary in the case of stem segments.

3.1 Calculating the Rotations

The amount of rotations in any of the three dimensional directions on a leaf is determined by several main factors:

- The rigidity of the leaf,
- The magnitude of the wind,
- The direction of the wind, and
- The length of the leaf segments.

Firstly, the effect of the wind force is hindered by the rigidity of the leaf. However, it is also important to note that the current orientation of a particular segment will determine the effective wind direction for that segment. This consequently leads to each segment requiring to store information of the effective wind as it will vary from segment to segment. The direction and magnitude of the effective wind force is governed by Algorithm 1.

Algorithm 1 Calculate Effective Wind Components

Variables
segmentTransMat = The transformation matrix of the segment orientation.
windDirect = The global wind direction.
effectWindDirect = The effective wind direction.
effectWindMag = The effective wind magnitude.

for each *segment* **do**
  segmentTransMat = convert(*segmentOrientation*)
  effectWindDirect = transform(*windDirect*, *segmentTransMat*)
  $effectWindMag = \frac{|effectWindDirect|}{leafRigidity}$
end for

Once the effective wind direction and magnitude is calculated, it is then possible to determine the effective rotations on the segment caused by the wind. The rotations are calculated in two stages. The first stage is to calculate the pitch and yaw rotations. The second stage is to calculate the roll rotation. Both stages involve calculating rotations for each of the segments in the leaf.

Note that all the rotations are calculated on the segments' local coordinate frames. Each of the segments is transformed so that the origin of the segment (which is at the bottom of the segment) matches the origin of the local coordinate frame. This means that the point of rotation is at the origin of each segment.

3.1.1 Pitch and Yaw Rotations

To calculate the pitch and yaw rotations, the *target axis* of the segment is set to lie along the length of the segment. The purpose of the target axis is to model the effective point of contact of the wind which is assumed to be the middle of the segment (shown in Figure 5). Hence the target axis length becomes dependent on the segment length. Since the target axis is along the length segment (z direction), the segment can then be rotated in the x and y directions, ie, yaw and pitch rotations.

The pitch and yaw rotations are calculated from the target axis and the effective wind direction and magnitude, or *effective wind vector*. The resultant of the sum of these two vectors represents the new direction of the target axis due to the wind as shown in Figure 6. In order to use this new direction, the quaternion that represents the rotation from the old to the new target axis is needed. The rotation axis and angle required for this quaternion is calculated using Algorithm 2.

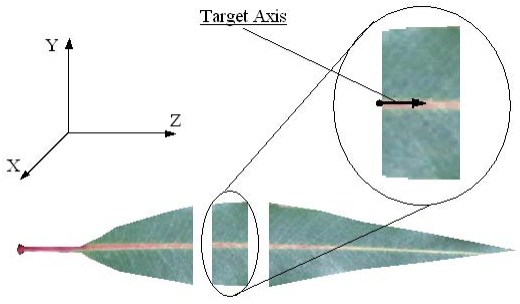

Figure 5: The initial orientation of the target axis during the calculation for the pitch and yaw rotations. Note that the target axis extends from the origin to the centre of the leaf segment.

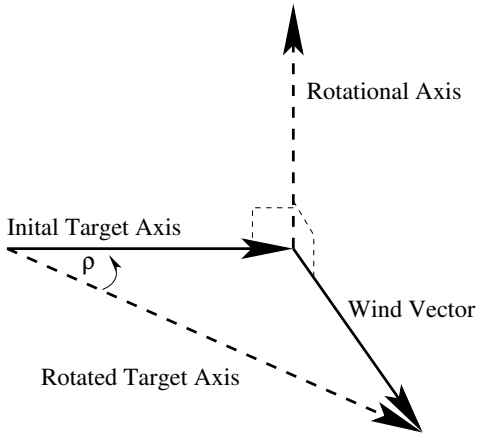

Figure 6: The rotation of the target axis due to the effective wind vector. The rotation angle ρ is calculated by the dot product of the initial and rotated target axes. The rotational axis is calculated through the cross product of the initial target axis and the wind vector.

With the rotational axis and angle ρ, the appropriate quaternion for representing the rotation is obtained for each segment. This rotation includes the components of both pitch and yaw.

3.1.2 Roll Rotations

The second stage of calculating the rotations of the segments involves dealing with roll rotations. It is found that the natural response of a leaf in nature is to try to align the surface normal of the leaf to directly oppose the wind direction. In other words, the leaf will roll so that it has the largest surface area facing the wind. This can be explained by the leaf's tendency to bend along the direction of the wind. Consequently, the leaf can bend the most when the largest surface area is facing the wind.

In roll rotations of the leaf segments, the rotational axis lies along the length of the segment. Therefore, to use the previous target axis orientation as for the pitch and yaw rotations would cause ambiguity (since the axes are parallel). Hence to avoid this problem, the target axis for calculating the roll rotations is set to be parallel to the normal of the leaf segment.

Note that since the leaf turns 'towards' the wind, the target axis is highly dependent on the wind direction. The target axis is either set to the positive or negative surface normal of the segment. To emulate the effect of the segment turning into the wind, the target axis is set to positive if the wind direction has a positive y component and to negative if the wind direction has a negative y component (shown in Figure 7).

Algorithm 2 Calculate Pitch & Yaw

Calculate Rotational Axis and Angle
rotataionAxis = crossProduct(*targetAxis*, *effectiveWind*)
Normalise(*rotationAxis*)
ρ = arccos $\frac{dotProduct(targetAxis,(rotatedTargetAxis))}{(|targetAxis|*|rotatedTargetAxis|)}$

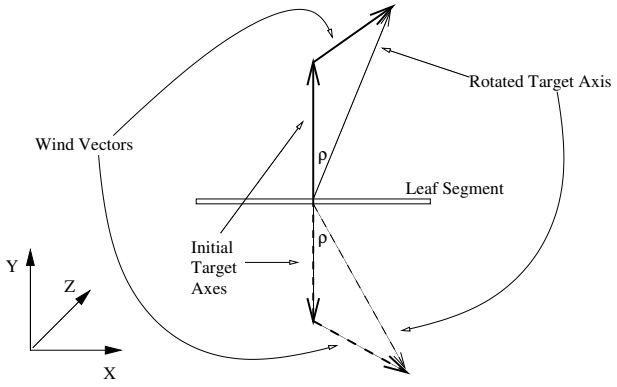

Figure 7: The rotations of the target axis due to the effective wind vector. Note that there are two sets of target axes. The appropriate one is chosen based on the y component of the wind vector. The rotations axis which is not shown is in the z direction and the angle of rotation ρ is calculated using the dot product of the initial target axis and wind vector.

Since we only need to deal with roll rotations at this stage, the z component (which causes pitch rotations) of the wind direction is removed. The roll rotations are then calculated via Algorithm 3, which is similar to the yaw and pitch calculations.

Algorithm 3 Calculate Roll

Set Wind Parameters
Set z parameter of *effectiveWind* to 0

Set the Target Axis
if *effectiveWind.y* > 0 **then**
  targetAxis = *segmentNormal*
else
  targetAxis = *-segmentNormal*
end if

Calculate Rotational Axis and Angle
rotationAxis = crossProduct(*targetAxis*, *effectiveWind*)
Normalise(*rotationAxis*)
ρ = arccos $\frac{dotProduct(targetAxis,(rotatedTargetAxis))}{|targetAxis|*|rotatedTargetAxis|}$

Once again, with the rotational axis and angle ρ, it is then possible to represent the roll rotation of each segment by these quaternions.

3.1.3 Combining Pitch, Yaw, Roll Rotations

With the two quaternions calculated previously (one for pitch and yaw, and the other for roll), it is then a simple matter of multiplying

them to combine the rotations. The resultant quaternion will then contain one rotational axis and angle that is equivalent of the three different components of pitch, yaw, and roll rotations combined.

3.2 Applying the Rotations

After the process of calculating the rotations for all the segments, each segment should now contain one quaternion that represents the combined rotations of pitch, yaw, and roll. This quaternion is then converted to a transformation matrix that is used to rotate each of the tagged vertices of the leaf model belonging to that particular segment. This causes appropriate vertices in the model to be transformed.

Since each segment is rotated in a local coordinate frame, each segment will also need to keep track of the collective rotations of all previous segments. This is done by a tracker transformation matrix that accumulates all of the previous rotations. Effectively, this tracker matrix is used to transform the previously local coordinates of the segments to global coordinates of the leaf.

Note that this tracker transformation matrix is also used to transform the global wind vector to be correctly aligned to the local segment coordinate frame. The whole process of converting the wind vector to local coordinates and how it is used in rotating the segment is illustrated in Figure 8.

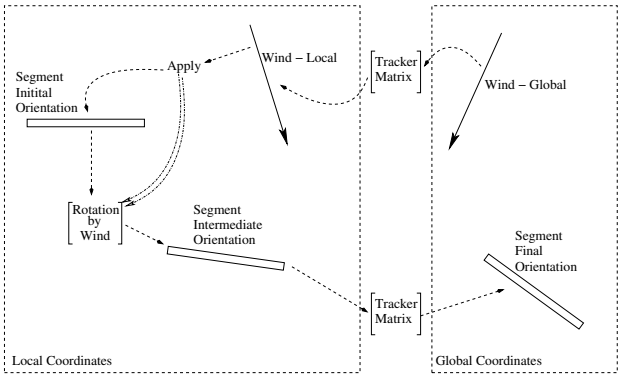

Figure 8: The procedure of rotating a segment. Firstly, the global wind vector is transformed into local coordinates to match the initial segment coordinate frame. This is achieved through the use of the *Tracker Matrix*. The segment is then rotated by the local wind vector through the *Rotation by Wind* matrix to reach the intermediate segment orientation. The segment is then finally transformed to global coordinates.

4 Restoration and Oscillations Forces

When a plant stem or leaf is bent by an external force, it stores potential energy. When the external force is reduced, or completely removed, the stem or leaf will 'spring' back into its original shape. This behaviour is modeled through the concept of a *restoration force*. Since the stem and leaves are treated in the same way, only the stem will be mentioned with leaves implied.

Once a stem is moving towards its original shape, the amount of kinetic force will increase and drive the stem to go beyond its original shape. It then reverses direction in an attempt to return to the original shape. This may happen several times, where the stem overshoots its goal and hence causes an oscillation effect. This behaviour is modeled by the inclusion of an *oscillation force*.

Both the restoration and oscillation forces are modeled to produce movement that mimics the movements of plants in nature. To implement this behaviour, we found it beneficial to define different modes for the stem. During the initial restoration of the stem to its original shape, the stem is in *restoration mode* (see Section 4.1). Once it reaches its original shape, the stem switches to *oscillation mode* (see Section 4.2).

Each of the stem segments has its own restoration and oscillation modes. This allows for the possibility that one segment can be in restoration mode while another segment is in oscillation mode. Also, both the restoration and oscillation forces are dealt with independently in each of the three dimensions. This allows for the possibility that a stem can oscillate in different directions at different times.

4.1 Restoration Mode

The restoration mode is activated when an external force (such as wind) is cut off, leaving the plant suddenly without external forces acting upon it. Perhaps a suitable scenario is a window that is suddenly closed, stopping the wind.

Without an external force bending the plant, a restoration force acts to release the stored potential energy. The magnitude of this restoration force is based on two major factors:

1. The rigidity of the stem, and

2. The magnitude of the previous wind force.

Both of these factors are intrinsic to how the restoration force will act. Firstly, the initial magnitude of the restoration force will be comparatively small and is based on the magnitude of the previous wind force. As time progresses, the restoration force increases linearly based on the rigidity of the stem as the potential energy is converted into kinetic energy.

Therefore, a stem that was deformed by a large external force will have a larger starting magnitude for the restoration force. Also, a stem with a high rigidity will have a restoration force that increases more rapidly than the restoration force of a stem with low rigidity.

The direction of the restoration force is perpendicular to the stem segments. This will allow the stem to return to its original shape in the most efficient way.

4.2 Oscillation Mode

At the point the stem reaches the position of its original shape, the stem will shift into oscillation mode. The oscillation force in this mode will cause the stem to oscillate past its original shape several times. Similarly to the restoration, the oscillation force is also based on two factors:

1. The rigidity of the stem, and

2. The magnitude of the last restoration force just before switching to oscillation mode.

The magnitude of the last restoration force will determine the magnitude of the oscillations. The greater the restoration force, the bigger the oscillations. The rigidity on the other hand is integral in determining the behaviour of the oscillations since the rigidity will determine the *speed* and the *total time* of the oscillations.

The function chosen to model the oscillations is based on a dampened cosine function(shown in Figure 9). The oscillation force follows the form:

$$oF = (rF - t * rF/totalTime) * \cos(t/speed)$$

Where:

oF = oscillation force.
rF = restoration force.
t = time of animation.
totalTime = total time of the oscillations, based on rigidity.
speed = speed of oscillation, based on rigidity.

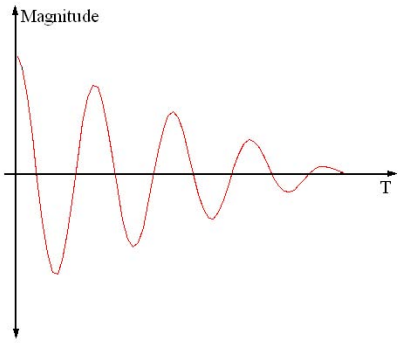

Figure 9: The dampened cosine function that is used to model the oscillations of the stem/leaf. The initial magnitude is determined by the restoration force. The speed and total time of the oscillations are based on rigidity.

5 Results

Throughout the testing of our approach to animate non-rigid plants, we used a small plant model with 20 leaves that were randomly arranged at runtime. The stem and each of the leaf models were segmented into 20 segments. The plant model was created using the *Anim8or* modeling program [Glanville 2003] and was textured with the help of the *UVmapper* program [Cox 2003]. The model (shown in Figure 10) consists of:

- 20 leaf models each with approximately 200 vertices,
- a stem model with approximately 500 vertices, and
- a pot model with approximately 450 vertices.

Figure 10: The plant model used during the testing of our approach. There are no external forces other than gravity at this point.

The machine which was used during testing has the following specification:

- Linux RedHat 8,
- Pentium IV - 2 GHz,
- 512MB DDR RAM, and
- Geforce 4 Ti4200 128MB.

5.1 Movements

To test the movement of the plant as a whole, we included user controlled wind forces. The wind can be increased in any of the three dimensions. Examples of wind blowing in the x direction in shown in Figure 11. The effects of a more complex wind direction is shown in Figure 12, where the y component is varied. Note that it requires a lot less change in magnitude in the y component to cause a large effect. This is attributed to the plant model's low rigidity in the y direction.

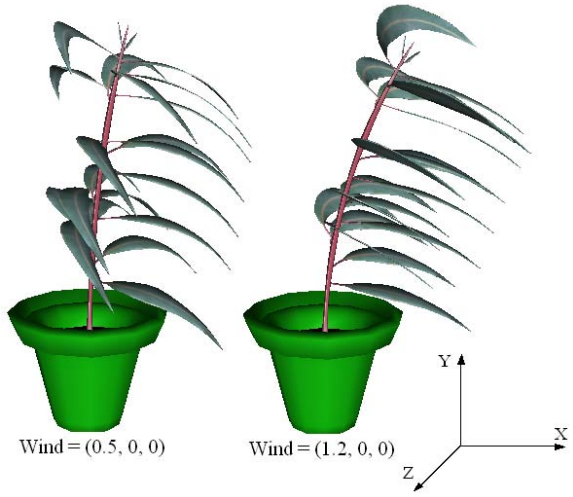

Figure 11: The effects of varying the x component of the wind from 0.5 to 1.2.

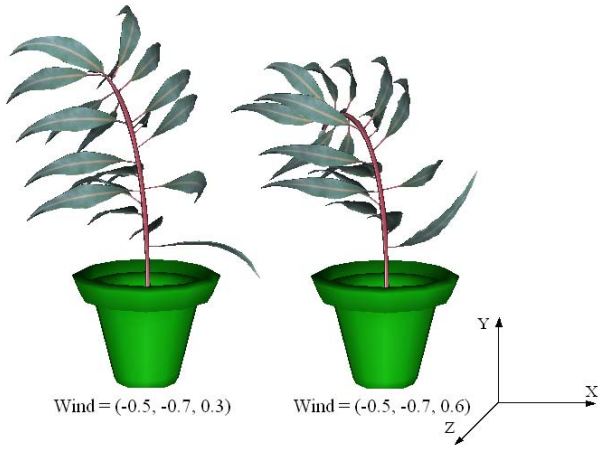

Figure 12: The effects of varying the y component of the wind from 0.3 to 0.6. Note that the x and z are kept constant.

To see an animated example of the restoration and oscillation modes, please refer to

http://www.csse.uwa.edu.au/~jasonw/videos/oscil.mpg. Screenshots of the animation are shown in Figure 13.

5.1.1 Randomised Wind

To further investigate the movements of the plant model, we introduced a randomised wind. This randomised wind is intended to mimic the inconsistent and unpredictable nature of wind.

Since the wind is defined by a three dimensional vector, the randomisation is achieved by simply randomly increasing or decreasing the wind vector components. To prevent large changes, the randomised wind only increments/decrements components by very small steps.

To see an animated example of the effect of a randomised wind, please refer to
http://www.csse.uwa.edu.au/~jasonw/videos/randW.mpg. Screenshots of the animation are shown in Figures 14.

5.2 Framerate

The framerate of the animation was found to run stably at approximately 34 frames per second. This framerate is consistent regardless of the magnitude or direction of wind applied to the model.

We conducted several experiments on another computer with a similar environment and CPU power. The only major difference is that it used an older Geforce 2 than the Geforce 4 used in the main testing machine. We found that the framerate was still around 32 frames per second. This confirmed the reliance on the CPU and not on the graphics card, as we do not yet take advantage of any hardware acceleration.

6 Future Work

In the current implementation, all the vertices in the models are transformed by the CPU. Optimizing this process will be essential in extending this method to multiple plant models. Possible optimizations include representing each segment by a bounding box and then using extrapolation techniques to fill in the rest of the segment. Also, the whole process of calculating and applying the rotations to each segment could be re-written as a vertex shader program to make use of possible hardware accelerations.

Only the effects of wind and gravity are currently accounted for. Future developments will include investigation into the inclusion of other external forces such as rain. Investigation will also be conducted on the effects of single point contact forces such as pushing the plant in the centre of the stem.

The positions of the leaves in the plant model are mainly manually manipulated, although there is some randomisation in the leaf orientations. Investigations will be conducted on providing a robust algorithm to produce realistically arranged plants. Algorithms involving Lindenmayer-Systems will also be considered [Prusinkiewicz et al. 1988].

7 Conclusion

In this paper, we propose a method for realistic rendering and animating small plant models in real-time. To allow for a large range of possible plant models which may include different species, we include a rigidity parameter. This rigidity parameter is used to determine the overall behaviour of the plant model. In this way, thin soft plants can be modeled through a low rigidity parameter whereas a stout rigid plant can be modeled through a high rigidity parameter.

The movements of the plant model are calculated through a simple system of force application. This allows for the animations to achieve suitable real-time framerates as well as provide realistic movements. Although these movements may not be physically accurate, they are however, visually realistic.

This method for modeling plants will be suitable for animating foliage in real-time applications. Future investigations will be conducted to optimize and extend this method to be able to handle several hundred models to create a vast realistically animated landscape of vegetation and foliage.

References

AITKEN, M., AND PRESTON, M. 2003. Grove: a production-optimised foliage generator for "The Lord of the Rings: The Two Towers". In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ACM Press, 37–38.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 604–611.

COX, S., 2003. Uvmapper. Available: http://www.uvmapper.com.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. 2002. Interactive visualization of complex plant ecosystems. In *Proceedings of the conference on Visualization '02*, 219–226.

GLANVILLE, R., 2003. Anim8or. Available: http://www.anim8or.com.

GUERRAZ, S., PERBET, F., RAULO, D., FAURE, F., AND CANI, M.-P. 2003. A procedural approach to animate interactive natural sceneries. In *CASA03*.

LOWE, N., STRAUSS, J., S.YEATES, AND HOLDEN, E. 2002. Auslan jam: A graphical sign language display system. In *Proceedings of the 6th Annual Conference on Digital Image Computing Techniques and Applications*, 98–103.

MILLER, G. S. P. 1988. The motion dynamics of snakes and worms. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, 169–173.

NOSER, H., RUDOLPH, S., AND STUCKI, P. 2001. Physics-enhanced L-systems. In *WSCG 2001 Conference Proceedings*, V. Skala, Ed.

POWER, J. L., BRUSH, A. J. B., PRUSINKIEWICZ, P., AND SALESIN, D. H. 1999. Interactive arrangement of botanical L-system models. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM Press, 175–182.

PRUSINKIEWICZ, P., AND MNDERMANN, L. 2001. The use of positional information in the modeling of plants. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, 289–300.

PRUSINKIEWICZ, P., LINDENMAYER, A., AND HANAN, J. 1988. Development models of herbaceous plants for computer imagery purposes. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, 141–150.

PRUSINKIEWICZ, P., HAMMEL, M. S., AND MJOLSNESS, E. 1993. Animation of plant development. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, 351–360.

RAMAMOORTHI, R., AND BARR, A. 1997. Fast construction of accurate quaternion splines. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 287–292.

SAKAGUCHI, T., AND OHYA, J. 1999. Modeling and animation of botanical trees for interactive virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, 139–146.

WEBER, J., AND PENN, J. 1995. Creation and rendering of realistic trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, 119–128.

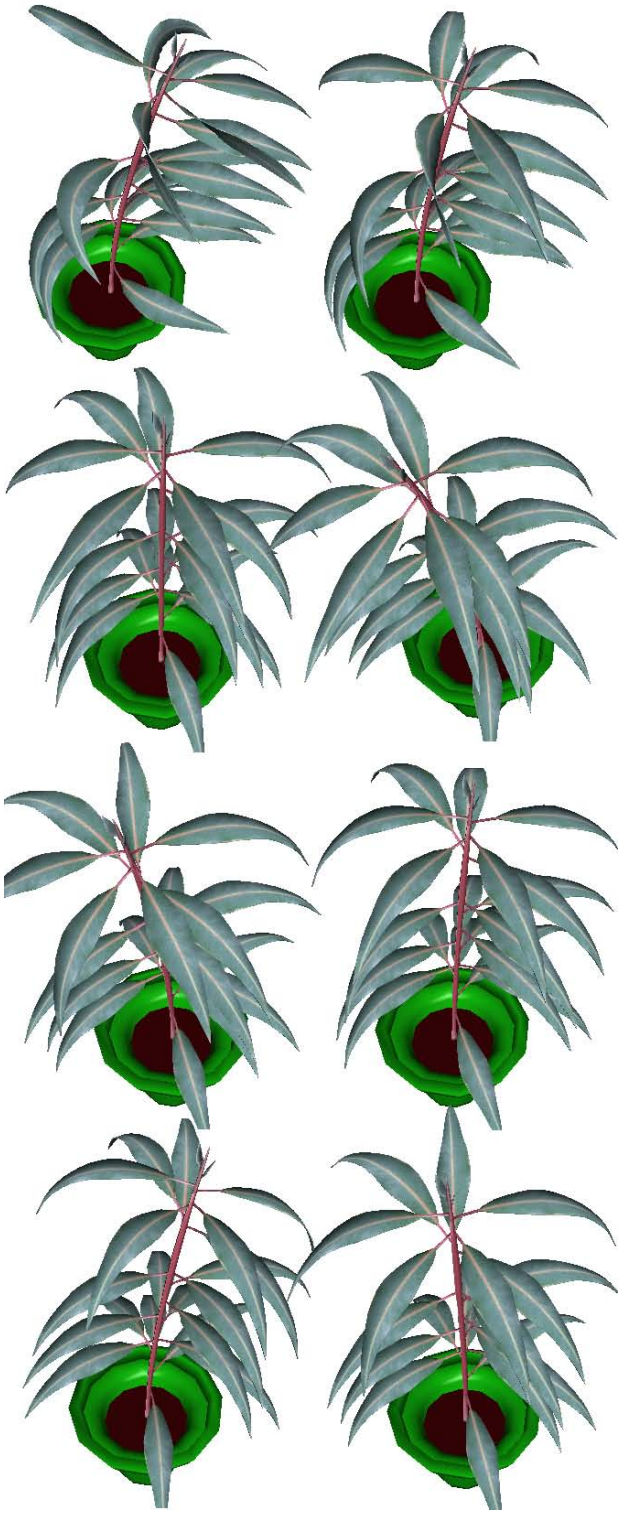

Figure 13: Some sequentially selected screenshots from the restoration and oscillation animation. The images go from left to right and top to bottom.

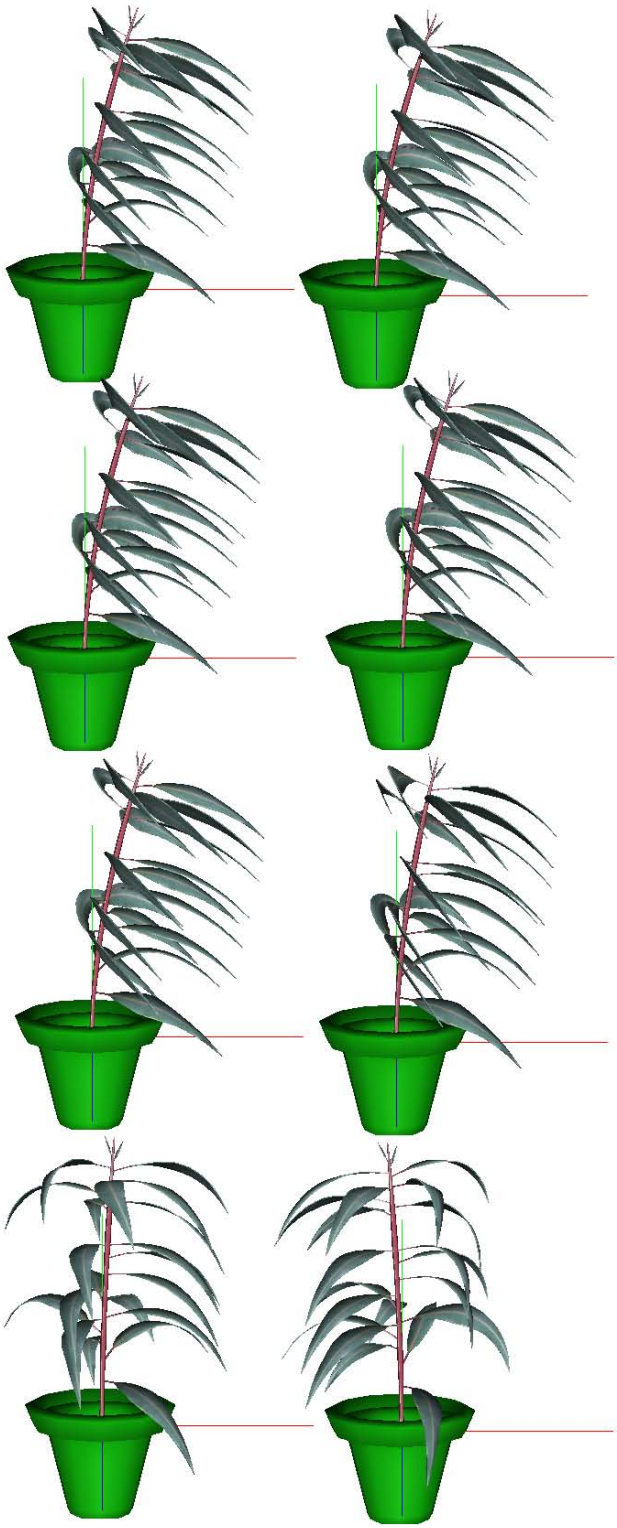

Figure 14: Some sequentially selected screenshots from the randomised wind animation. The images go from left to right and top to bottom.

WEISSTEIN, E. W., 1999. Quaternion – from mathworld. Available: http://mathworld.wolfram.com/Quaternion.html.

WEISSTEIN, E. W., 2003. Cantilever – from eric weisstein's world of physics. Available: http://scienceworld.wolfram.com/physics/Cantilever.html.