Real-time Animation of Underbrush

Luis Carlos Yano Endo, Carlos Hitoshi Morimoto, Antonio Elias Fabris
Computer Graphics and Applied Computational Geometry Project - CGCAP
Instituto de Matemática e Estatística, Universidade de São Paulo
Cidade Universitária - Rua do Matão, 1010, Caixa Postal 66281, 05315-970, São Paulo - SP, Brazil
{lye,hitoshi,aef}@ime.usp.br

ABSTRACT

Modeling natural phenomena is a problem that has been studied for a long time in Computer Graphics. A problem that currently arises is how to render and animate realistically natural phenomena in real-time. This paper presents a technique to model underbrush with fast and realistic animation. Pre-computations and a simplified physically based model were developed to achieve the work's proposal.

Keywords
Modeling and Rendering of Natural Phenomena, Physically Based Graphics, Animation with Constraints, Real-time Rendering.

1 Introduction

Real-time generation and animation of complex tri-dimensional scenes is still considered a difficult task, but it has become a requirement for simulators, games and virtual reality applications. In particular, modeling realistic natural phenomena requires a great number of polygons, thus requiring a great number of calculations for rendering.

In this paper we present a computer graphics framework to create animations of realistic natural scenes with underbrush in real-time. The framework allows to model physical and ecological features for different kinds of underbrush, such as grass and flowers. Although these plants have relatively simple geometry, the number of polygons become very large due to the number of instances of these plants in the scene.

Many control properties must be considered in order to create a flexible system to model a great variety of underbrush. Some physical properties like mass, width and thickness of the stalk of leaves will be as important to the appearance of the plants as their color and texture. The model also considers external properties of plants. For example, the concentration of nutrients on the ground is used to define the underbrush distribution over the terrain.

We use a simplified physically based model [Wit97a] in order to achieve real-time and realistic results. The leaves and branches of the underbrush are bent due to the force of gravity. An opposing elastic force is created to compensate the force of gravity until an equilibrium state is reached. Pre-computations of these states, that define the rest shape of the plants, are used to render the scene in real-time.

The framework allows to model wind that animates the underbrush. It can also be easily extended to generate animation by the action of any external element that can apply forces on the plants, such as rain and a soccer ball for example. Realistic scene animation is obtained from the computation of the resultant of these forces, using a Dynamic Constraint Model. The direction and strength of the forces define the velocity of the plants movements, producing very realistic animations. The computations are simplified using some natural properties of the plants.

The next section describes some advantages and problems of other techniques used for real-time realistic animation of underbrush. The first step of the framework is to create a simple physically based model of underbrush, which is described in Section 3. These models can be animated using Dynamic Constraints and force fields as explained in Section 4. Section 5 describes the algorithms used to distribute the vegetation and compute the level of details of the scene. Section 6 presents some experimental results, and Section 7 discusses advantages and limitations of this framework, and future work.

WSCG SHORT PAPERS proceedings, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.
Copyright UNION Agency - Science Press

2 Fast and Realistic Underbrush Animation

Real-time animation of realistic scenes is still a hard problem due to the balance between performance required for real-time animation and the level of detail required for realistic rendering.

Several modeling techniques were proposed to achieve realistic rendering. Models based on Bézier curves and surfaces are presented in [Deu98a, Fow92a]. These methods are appropriate to render scenes for any viewing situation, even when the plants are very close to the viewer. Reeves [Ree85a] proposes a different technique for plant modeling that is also appropriate to render realistic scenes. His method is based on particle systems, where a blade of grass and other underbrush are represented by the trajectory of a particle. Unfortunately, both these methods are very computationally expensive, thus inappropriate for real-time animation.

Multi-level representations play an important role on the generation of real-time animations of complex scenes. The idea is simple: detailed images, with thousands of polygons, are generated for of objects close to the viewer, while rendering is performed using less polygons for objects beyond a certain distance, due to limitations of the human eye and the output device. For example, Perbet and Cani[Per01a] use three levels of detail to achieve real-time animation of grass fields under windy conditions. Similar techniques are also used by Di Giacomo *et al.*[Gia01a] for animating and rendering a forest and Markosian *et al.*[Mar00a] for the creation of grassy landscapes, trees or furry creatures using hand-drawn representations.

Image based and Point based techniques[Deu02a] have been used for efficiently rendering distant instances of plants. However, it would be very unrealistic to use these for rendering mid distances plants, besides realistic animation techniques for these modeling could not be found yet.

Our framework implements a single representation appropriated for realistic generation of underbrush from mid distances, but that can be easily extended to multiple levels. The modeling technique is based on vectors under the action of force fields. Vectors are used to represent the shape of underbrush and animations are obtained from the interaction of the force fields with these representation vectors. This technique is described in detail in Section 3.

To achieve real-time performance during scene animation our system pre-computates the interactions so that the shape states can be retrieved instead of being computed for different states of the force fields. Using these pre-computed information for the calculation of the initial and final position of plants, we achieve realistic animation by using the Dynamic Constraints Model [Bar88a] to generate the movement of underbrush from the initial to the final position defined. This technique already proves to create realistic animation in [Met92a], for the animation of physics models.

Wejchert and Haumann [Wej91a] propose an aerodynamics based method to simulate and control the motion of objects in fluid flows. This method was used in [Per01a] to animate prairies. In Section 4 we describe a simpler technique, derivated from this one, that allows not only animation by the action of fluids, but real-time animation caused by any other external agent that can apply a force on the plant.

3 Modeling Underbrush

This section describes the model used in our framework. Real-time requires simple models that can be computed efficiently, while realism requires complex models that are hard to compute. We achieve good realism and performance by simplifying physically based models of plants and using pre-computations.

3.1 Pre-computations and Data Structure

Calculations must be avoided during the animation in real-time systems. In scenes, such as underbrush, with thousands of instances of different plants, each calculation spared will improve the animation performance. Since underbrush have simple geometry and many instances are similar to each other, calculations made for one of the plants may be used for the others. In this section, we describe a data structure that allows the efficient storage of this physical information and that can be used to model different plants.

Underbrush is composed by a combination of branches, leaves and flowers. Every leaf or flower is fixed to an object, that can be the ground or a branch. All objects are modeled using polygons represented by vectors instead of points. Vectors are structures that enables easy and fast changes in the shape of plants. To create a new shape component we only need to add the vectors that defines its shape to a initial point, that is the point where the component is fixed.

Underbrush components are defined by three sets of vectors. The first set defines the main axis of the component. The other two sets define the right and left borders of the component. These border vectors are perpendicular to the main axis, and the shape of the component is defined by connecting the ends of the border vectors. Figure 1 show how a leaf and a branch are represented.

Using these set of vectors, we can pre-computate some components shape including the calculation of

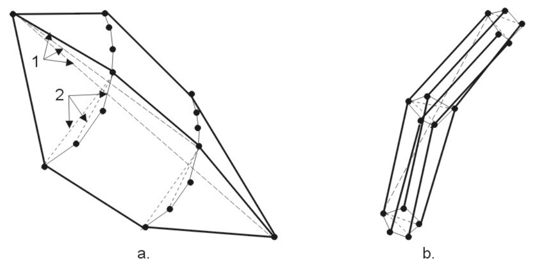

Figure 1: (a) A simple leaf in vector representation, (a.1) the center vector set of this leaf, (a.2) shows the right vector set. (b) A branch in vector representation. Flowers are represented using the same technique applied on their petals.

physical properties which is shown in next subsection. The shapes are pre-computed and stored for different angles of the component with the terrain normal. The numbers of angles for the pre calculation may be defined by the system's user. Animation is performed applying a force on a component, who has an initial angle ρ with the normal, that will bent to an angle ρ', since the vectors sets for this angle have been pre-computated, the animation is performed very fast.

Hundreds of plants, at a mid distance from the viewer, can be modeled using this technique. Plants closer to the viewer may be modeled using a more realistic technique, that can use more vectors to define the plant shape and do not use pre-computations to avoid similar movements, or even use the points defined by the vectors sets as control points for Bézier surfaces. The animation time would be similar since plants closer to the viewer cover many other plants far from the viewer.

3.2 Physical Model of Underbrush

This section describes how to model underbrush in its rest position, ie without the action of other external forces but gravity. The next section shows how to animate the plant considering the action of other external forces.

Our model considers only two forces, the force of gravity ($\vec{F}_g$), that pulls down the component (c_u) of the underbrush towards the ground, and an elastic force ($\vec{F}_{el}$), that opposes the action of the force of gravity. The force of gravity depends only on the mass of the component. The elastic force is proportional to the component's displacement from its original position, ie without action of gravity. The rest position of c_u is obtained by the equilibrium of these two forces.

The algorithm for component bending will divide it in sections. Each section, will be bent by the action of different resulting forces, considering the action of $\vec{F}_g$ and $\vec{F}_{el}$. While c_u is being bent the sections already bent are fixed for simplification. The first section bent is the closest to the its branch or root, since it is already fixed.

$\vec{F}_g$ bends the component at the section fixed, and, as the it bends, $\vec{F}_{el}$ increases. This elastic force has a linear dependence on the distance from the point $\vec{X}$ to $\vec{X}_0$, the center of mass of the sections above the one that is being bent after and before the bending. This force also depends on an elastic constant k, defined for each section as a function of some component properties like the thickness of it's stalk and c_u 's width and section position.

These two forces can be decomposed in an component normal to c_u, and a component along the direction of sections that are not bent yet, as we can see in Figure 2.

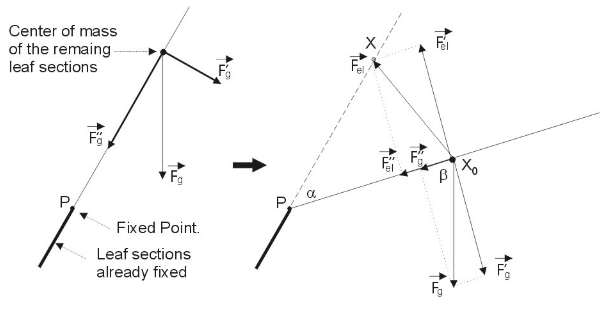

Figure 2: The section fixed and all components of $\vec{F}_g$ and $\vec{F}_{el}$ when the normal components of the forces are equals. In this figure the elastic force is equal to the difference $\vec{X}_0 - \vec{X}$ just to facilitate the visualization, the elastic force is described by $\vec{F}_{el} = k.(\vec{X}_0 - \vec{X})$.

The reaction for component along the direction of sections that were not bent yet, is on the terrain, since we fixed all sections bent, so we do not need to handle it. When the normal component of $\vec{F}_g$ and $\vec{F}_{el}$ are equal, an equilibrium is reached and c_u stops to bend. The position of equilibrium is stored by the currently bending section, the next section is fixed. The same calculation must be done to all the remaining sections. At the end of this process, c_u is bent by the action of $\vec{F}_g$. A simulation of this process can be seen in Figure 3.

Figure 2 shows the component bending when the component normal of $\vec{F}_g$ and $\vec{F}_{el}$ are equal. Given α the angle of bending and β the angle between the component in the equilibrium position and $\vec{F}_g$, we can easily write an equation to discover the angle α that we need to bend c_u.

$$\vec{F}'_{el} = \vec{F}'_g$$

$$k.(X_0 - X).\sin\frac{180-\alpha}{2} = m.g.\sin\beta$$

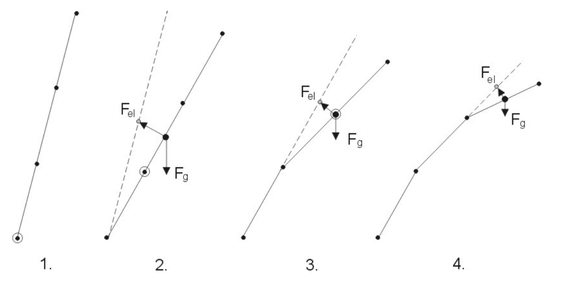

Figure 3: The bending of a leaf or branch scheme due $\vec{F_g}$: (1) The leaf before the bending calculations. (2) The leaf already bent on the first point by the influence of $\vec{F_g}$ and $\vec{F_{el}}$ opposing to the component bending. (3,4) The same calculation for the further points is shown.

The angle we must bend c_u at each section is given by equation 1. Where γ is the angle between $\vec{XP}$ and $\vec{g}$.

$$\alpha = \arcsin \sqrt{\frac{(\sin\gamma)^2}{(\frac{k.(X_0-X)}{m.g} - \cos\gamma)^2 + (\sin\gamma)^2}} \quad (1)$$

Plants with different appearances can be created by changing the values of the mass, thickness and width of each c_u's section. Figure 4 shows images of different tufts of grass created using different values of the parameters described above.

If $\vec{F_g}$ reaches a value greater than a defined maximum value for $\vec{F_{el}}$, the component will ”break”. Figure 4.c illustrate the “broken leaf”.

4 Fast Physically Based Animation for Underbrush

The animation will also be created using physically based models to achieve realistic results. The underbrush will move only by the action of an external force over it. We will use force fields to simulate the action of forces applied to the objects in the field.

For the animation, other forces must be considered in the calculation of the equilibrium position of a component of the underbrush, these external forces will be defined by the force field. If the component is under effect of more than one field in the force field, a resultant external force is calculate following the Euler's principle of superposition, that allows us to combine the forces applied to a body into an equivalent force applied at the center of mass.

$$\vec{F} = \sum_{forces\ i} \vec{F_i}$$

To generate realistic animation, the Dynamic Constraints model will be used. Barzel[Bar88a] describes it using the following definition. ”Constraints forces are analog to the internal forces which hold the parts of compound objects together. Constraint forces also assemble the models, pulling their components into their proper configuration”.

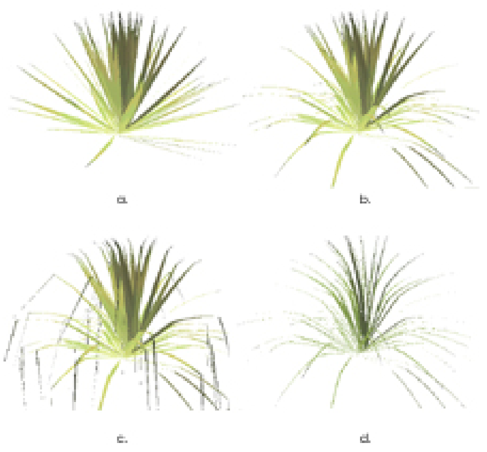

Figure 4: (a) A tuft of grass without physics properties. (b) A tuft of grass with physics properties, under action of $\vec{F_g}$ and $\vec{F_{el}}$. (c) A tuft of grass with broken leaves. (d) A tuft of grass with physics properties but with different features (parameters) from a, b and c.

As we assume that a component always has its base fixed on a branch or on the ground, the plants can not spin over its own base, so we do not need to concern about angular velocities or accelerations, and consequently about Torque, Angular Momentum and Inertia Tensor. This simplification avoid many unnecessary calculations.

Using constraints we can have an animation more realistic than just make an interpolation between the initial and final position. The leaves and branches will move with different velocities because the constraint force defines different accelerations during the animation.

4.1 Force fields over the Scene

The aerodynamics based method for animation of objects in fluid flows, presented in [Wej91a], defines flow primitives that represents velocity fields. These fields determine the direction and intensity of the fluid. Objects or obstacles placed in the velocity field receive the action of a force, that depends on the area, position, and orientation of the object.

In order to simplify the calculations, we define a force field over the terrain. A fluid flow defined may change the force applied to objects in each the position of the force field, without the modeling of velocity fields. Besides the gain in performance, this model allows the animation to be performed by any other agents that can apply a force, including fluids flows.

The force field is a volumetric mesh of forces defined above the terrain, each part of the mesh defines a volume. Objects inside this volume receive the action of a resultant force defined for this element of the force field.

The plants are animated by the action of the resultant force defined by the volumes of the force field where the plant is. The algorithm for the animation will be described in next subsection. Figure 5 shows an example of the force vectors on the force field.

Wind flows can be modeled changing the force field intensity and direction. All the primitives shown in [Wej91a] can be created by changing directions and intensities of the force field vectors. The force propagate for the adjacent volume in each animation step, calculating the force propagated considering the interaction with some object in the field, that can possibly change the vector.

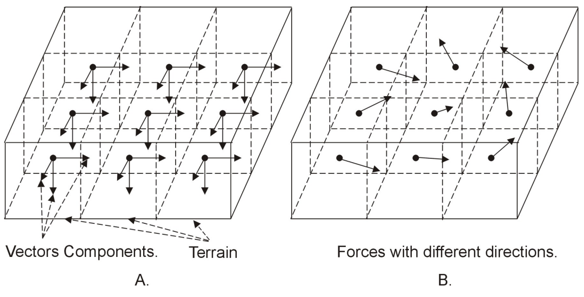

Figure 5: (a) The force field over the terrain. (b) An example of force vectors simulating a wind flow.

4.2 Fast Animation using Pre-Computed Physical Models

At each step of animation the components (c_u) must check the force defined by the force field to update their positions. The resultant force applied on a component is computed from force field, $\vec{F}_g$ and $\vec{F}_{el}$ that act on c_u. Each pre computed position of c_u will have a interval of forces norms associated, this can be used to find quickly the position the component the force is defined. c_u can be simply placed at the right position once the force vectors are available.

A new elastic force ($\vec{F}_{el_a}$) is defined in the animation. While the elastic force defined in the modeling, $\vec{F}_{el}$, was used to create a resistance for c_u bending due the action of the force of gravity, $\vec{F}_{el_a}$ opposes to the change of c_u's resting position, calculated on the modeling phase. Other important difference is that the animation is done for the entire component at once, though each component section might have many modeling elastic forces.

The animation elastic force exists only when some force changes the component resting position, and becomes stronger as the distance between the position of rest and the position caused by the external force increases. Figure 6 illustrates the behavior of $\vec{F}_{el_a}$.

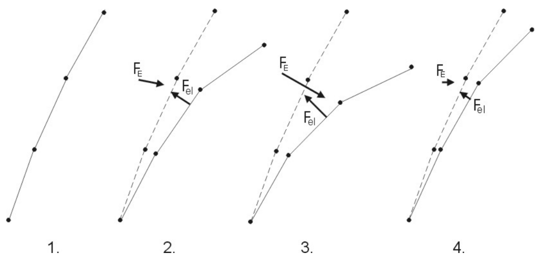

Figure 6: (1) The component in its resting position. (2) c_u receives the action of a external force $\vec{F}_E$, $\vec{F}_{el_a}$ tries to restore the component's initial position. (3) $\vec{F}_E$ increases and c_u bends more until the equilibrium between $\vec{F}_E$ and $\vec{F}_{el_a}$ is established. (4) $\vec{F}_E$ decreases so $\vec{F}_{el_a}$ take the component closer to its original position.

$\vec{F}_{el_a}$ is computed from the difference between the center of mass before and after the position change. When $\vec{F}_{el_a}$ becomes bigger than the external force ($\vec{F}_E$) applied, c_u tends to return to its resting position.

The animation is generated using morphing, an interpolation technique, for components closer to the viewer in order to achieve a smoother animation. To create variations in the velocity of the interpolation, we will use the Dynamic Constraint Model that is described on next subsection.

4.3 Realistic Animation using Dynamic Constraints

To create a more realistic animation for the underbrush, with non uniform velocities, we will use an approach of the Dynamic Constraint Model developed by Barzel and Barr [Bar88a], considering some simplifications the plant system may offers us.

In the Dynamic Constraint Model, the equation 2 is given to define a single constraint force calculation.

$$\sum_{constraints\ j} \Big(\sum_{bodies\ i} (\Gamma^i G^i_j + \Lambda^i H^i_j)\vec{F}_{e_j} \Big) + \sum_{bodies\ i} (\Gamma^i \vec{F}^i_E + \Lambda^i \vec{T}^i_E) + \vec{\beta} + \frac{2}{\tau}\vec{D}^{(1)} + \frac{1}{\tau^2}\vec{D} = 0 \qquad (2)$$

It is a very complex equation, but T, H and Λ are related to the torque of the body, in this system these variables are null. Our model is very similar to the Point - to - Nail constraint example 2 of Barzel and Barr's paper, this example illustrate a body under effect of gravity and without rotational terms. τ is a time constant that is used to control the time for the constraint to be satisfied.

Barzel and Barr's defines $\vec{D}$ as the distance from the current position of the body to the constraint point it may reach, $\vec{D}^{(1)}$ as the rate of change of $\vec{D}$ and $\vec{D}^{(2)}$ as the acceleration of $\vec{D}$. $\vec{D}^{(2)}$ is defined by equation 3.

$$\vec{D}^{(2)} = \sum_{bodies\ i} (\Gamma^i(y)\vec{F}^i + \Lambda^i(y)\vec{T}^i) + \vec{\beta}(y) \quad (3)$$

Let $\vec{X}$ be the position of the center of mass before the constraint is reached and $\vec{X}_0$ the position of the constraint point, the deviation $\vec{D}$, velocity $\vec{D}^{(1)}$ and acceleration $\vec{D}^{(2)}$ are given by the following equations:

$$\begin{aligned} \vec{D} &= \vec{X} - \vec{X}_0 \\ \vec{D}^{(1)} &= \vec{v} \\ \vec{D}^{(2)} &= \vec{F}/m \end{aligned} \quad (4)$$

From 3 and 4 we have $\beta = 0$ and $\Gamma = 1/m$. We only have the force of gravity acting over the body, so $\vec{F}_E = \vec{F}_g$.

This can give us the constraint force need to take the body from $\vec{X}$ to $\vec{X}_0$.

$$\vec{F}_c = -\vec{F}_E - \frac{2}{\tau} m\vec{v} - \frac{1}{\tau^2} m(\vec{X} - \vec{X}_0)$$

The Dynamic Constraint Model calculates the constraint force needed to take the body to the constraint point defined. In this case we can see that the constraint force may be decomposed in three components, one opposing the force of gravity, one opposing the body's velocity and other that pulls the body to the constraint point.

In our approach we will use the same model but instead of calculate the constraint force we will define a constraint force to get the object position when the forces reach the equilibrium. Since we are modeling plants, that can not move, the velocity component is null. Our problem is simplified to find $\vec{X}_0$ given $\vec{X}$, the force of gravity acting over the plant and a constraint force that will be defined considering some properties of the plant.

$$\vec{F}_c = -\vec{F}_E - \frac{1}{\tau^2} m(\vec{X} - \vec{X}_0) \quad (5)$$

For a time interval τ, we can create an animation using equation 5. The constraint force will be calculated when the center of mass of the leaf changes from $\vec{X}$ to $\vec{X}_0$, by single vectors operations.

We calculate the effects of the constraint force for subintervals of τ, and the position the leaf will be in each time interval. This creates a more realistic animation since the leaf will move under different velocities and accelerations.

5 Level of Detail and Distribution

After the computation of the physical properties that define the leaves, the creation of the points that will be part of the leaves is quite simple.

We can save some computer resource also using a simple level of detail technique to diminish the number of polygons required to generate the scene. A simple distribution algorithm was also developed, to simulate plants propagation and competition for space over the terrain.

5.1 Level of Detail Algorithm

If the leaf is a certain distance far from the viewer we can represent the grass leaf with less details. The number of triangles required to generate a good image for far leaves is smaller, so we represent the leaves using less triangles, what helps us to generate a scene in real-time.

The larger the field of view of the camera, the more instances of grass have to be rendered. Thus if fewer polygons are not used to represent a plant instance, it is very difficult to create a scene in real-time. A simple algorithm can solve this problem, if the camera is at a certain distance from the tufts, some vectors can be eliminated, defining less points to the grass leaves.

Using this simple algorithm we can reduce the number of polygons that will be rendered, what will help us to generate scenes in real-time. This technique can be found in the Java3d API [Sun02a].

The hierarchical technique using different models for the grass at different levels of detail is used in [Per01a], but this technique must handle the transition between models. To create more realistic results we intend to create one technique for rendering plants closer to the viewer, as was mentioned previous. Another level of detail, that can be a texture mapped over the terrain or a image based technique, will also be implemented, for distant underbrush rendering. This technique will be used for mid distances plants.

5.2 Underbrush Distribution

Some important properties are defined in the terrain, they determine the distribution of the underbrush and they are very important to the scene appearance.

We can think of the terrain properties as nutrients that the underbrush needs to grow. If we define a terrain with few nutrients we will obtain dispersed underbrush. Each kind of underbrush will use some nutrients of the terrain sections it occupy, so the number of instances will also be determined by the concentrations of nutrients over the terrain.

The distribution of nutrients is also important to determine where each kind of underbrush will grow (different kinds of vegetation requires different kinds of nutrients), and which terrain portion it will occupy.

A simple algorithm to generate the distribution of the underbrush over the terrain mesh has been developed. This algorithm receives a propagation point of the distribution, and from this point it computes where there will be new plants.

The algorithm verifies if the terrain has the required nutrients that are necessary for the underbrush specified to grow, and returns the points on the terrain that will be occupied by grass. Regions with great nutrients concentrations will be more populated by plants, that is, the algorithm directs propagation of plants to regions with more nutrients.

The distribution of grass tufts may be done by direct user specification or by procedural generation. This algorithm allows the definition of other kinds of underbrush that may compete for space and nutrients. In future work we intend to have other kinds of plants to compete for space, light and nutrients over the terrain. This kind of simulation is also shown in [Deu98a].

The distribution algorithm is part of the pre calculations of the system, and takes less than 3 seconds for a scene filled with plants instances.

6 Experimental Results

The system prototype was developed on a Pentium III 900Mhz with 512MB running Linux. We also use a 64MB graphics card to help us to generate real-time animations.

All the rendering algorithms were implemented using the Java3D API [Sun02a]. The Java platform was chosen for its stability, high-quality, scalability and platform independence. The Java3D API has implementations based on OpenGl or DirectX and allows easy creation of web applications.

After a few seconds of pre computation of the constraints and force fields, the animation created to simulate wind flow is generated in real-time using the technique of force fields. Figure 8 and 9 illustrate this animation. For both figures the underbrush start without any external force acting over it, as we can see in (a). A wind flow pass over the scene changing the plants position (b), when the wind force effects are reduced the plants are forced to return to its original position (c).

The framerate for rendering, for a resolution 500x500, is between 10 to 20 frames per second, depending on the number of instances shown and the movement of the user over the scene.

For grass fields the system was able to render 200 tufts of grass, in real time. In the flower field, we as able to render about 200 flowers with 1000 leaves, both allowing user's movement over the scene. This results proves to be very satisfactory for a mid distance representation.

7 Conclusion and Future Work

We presented a new technique for modeling underbrush using dynamic constraints. Real-time performance is achieved by pre computing some physical properties and states and by hierarchically modeling the leaves. The shape of the plants is created considering the effects of its weights and a constraint force that oppose to the leaf bending.

The framework pleases the modeling of a great variety of underbrush, and can be further extended to any kind of plant. We plan to extend the system to include the generation of bushes and trees, and to study some distribution algorithms for different kind of plants.

Further we intend to render plants closer to the viewer using splines or other interpolation and plants textures technique for realistic results on plants close up's. An image based algorithm or a point based technique may also be developed for plants far from the viewer, this may be required to achieve real-time results for an ecosystem simulator.

As we intend to continue to animate natural scene in real time, the next step is the realistic animation of bushes and trees. L-Systems [Pru96a] are known by its ability to model natural objects, particularly botanical and cellular models. Our task must be how to use L-System to generate fast plant models.

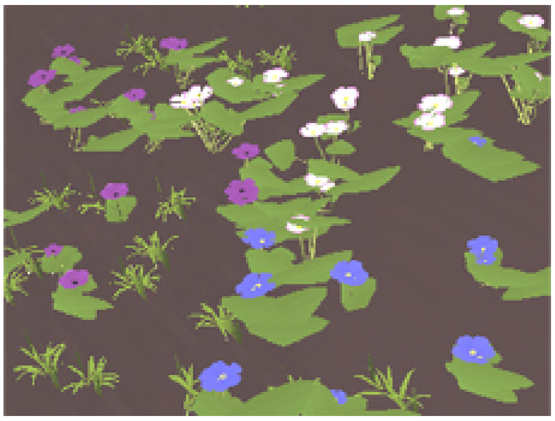

Figure 7: Underbrush scene animated in real time.

8 REFERENCES

[Sun02a] Sun Microsystems. Java 3D^{TM} API Tutorial.

[Bar88a] Barzel, R. and Barr, A. H. A Modeling System Based On Dynamic Constraints. In Computer Graphics 22, pp. 179-188, 1988.

[Deu02a] Deussen O., Colditz, C., Stamminger M., and Drettakis, G. Interactive visualization of complex plant ecosystems. In Proceedings of the IEEE Visualization Conference, 2002.

[Deu98a] Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr M., and Prusinkiewicz, P. Realistic Modeling and Rendering of Plant Ecosystems. In Computer Graphics 32, pp. 275-286, 1998.

[Fow92a] Fowler, D. R., Prusinkiewicz, P., and Battjes, J. A collision-based model of spiral phyllotaxis. In Computer Graphics 26, pp. 361-368, 1992.

[Gia01a] Giacomo, T. D., Capo, S., and Faure, F. An Interactive Forest. In Eurographics Workshop on Computer Animation and Simulation, pp. 65-74, 2001.

[Mar00a] Markosian, L., Meier, B., Kowalski, M.,Holden, L., Northrup, J., and Hughes, J. Art-based Rendering with Continuous Levels of Detail. In Proceedings of the First International Symposium on Non Photorealistic Animation and Rendering (NPAR) for Art and Entertainment, 2000.

[Met92a] Metaxas, D., and Terzopoulos, D. Dynamic Deformation of Solid Primitives with Constraints. In Computer Graphics 26, pp. 309-312, 1992.

[Per01a] Perbet, F., and Cani, M. P. Animating Prairies in Real-Time. In ACM Interactive 3D Graphics, 2001.

[Pru96a] Prusinkiewicz, P., Hammel, M., Hanan, J., and Měch, R. L-Systems: From the Theory to Visual Models of Plants, 1996.

[Ree85a] Reeves, W. T., and Blan, R. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. In Computer Graphics 19, pp. 313-322, 1985.

[Wej91a] Wejchert, J., and Haumann, D. Animation Aerodynamics. In Computer Graphics, pp. 19-22, 1991.

[Wit97a] Witkin, A., and Baraff, D. Physically Based Modeling: Principles and Practice. In SIGGRAPH'97 Course Notes.

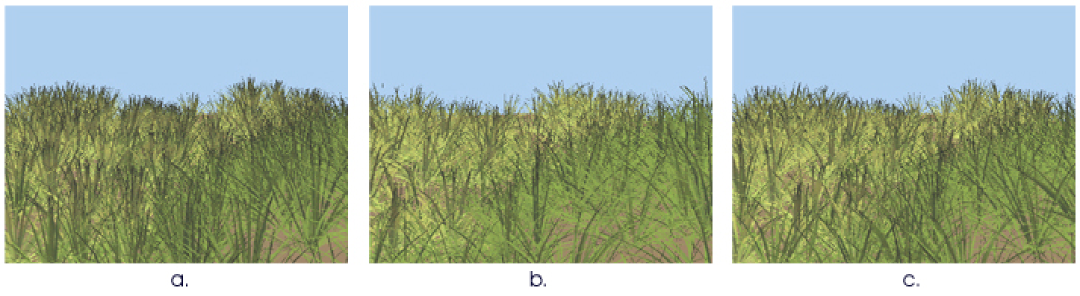

Figure 8: A grass field animation. The leaves bend when a wind flow pass over the scene.

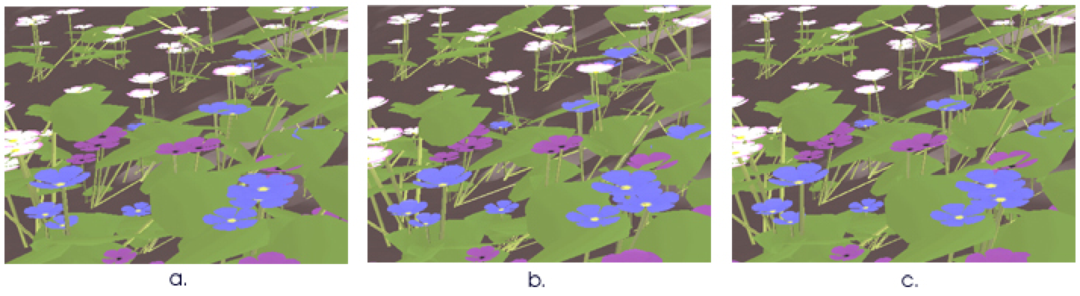

Figure 9: A field of flowers animated in real time by the action of the wind.