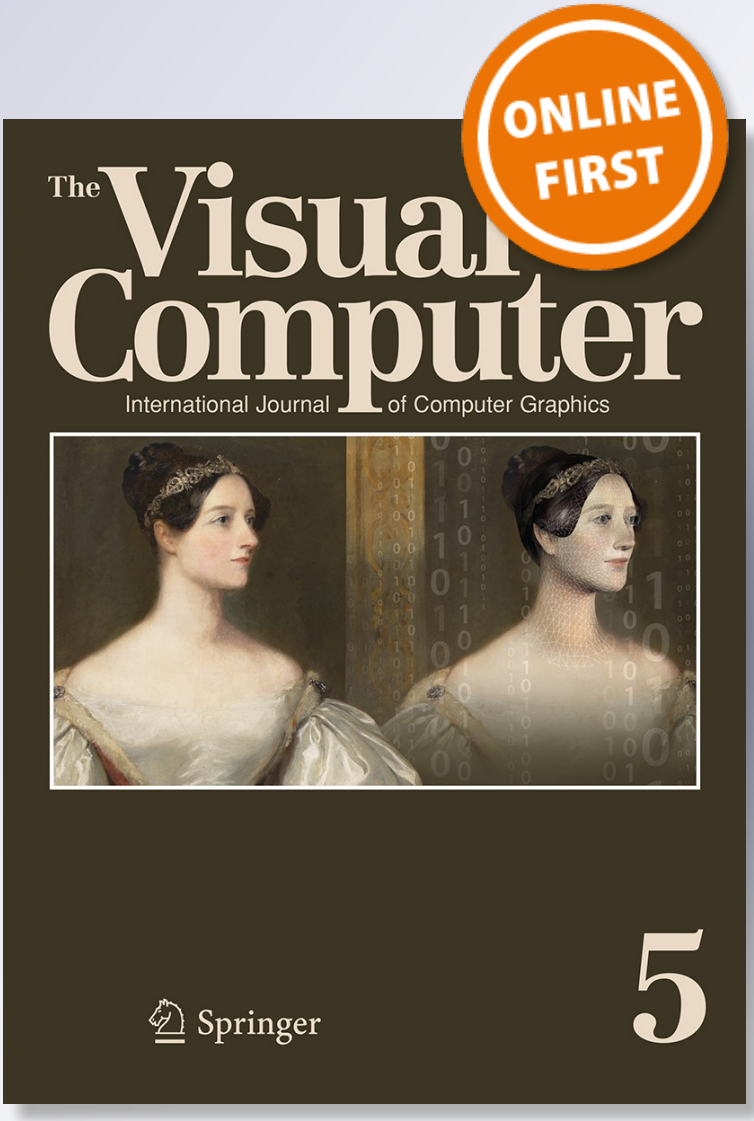
A simulation on grass swaying with dynamic wind force

Ruen-Rone Lee, Yi Lo, Hung-Kuo Chu & Chun-Fa Chang

The Visual Computer
International Journal of Computer Graphics

ISSN 0178-2789

Vis Comput
DOI 10.1007/s00371-016-1263-7



Springer

Vis Comput
DOI 10.1007/s00371-016-1263-7

ORIGINAL ARTICLE

A simulation on grass swaying with dynamic wind force

Ruen-Rone Lee[1] · **Yi Lo**[2] · **Hung-Kuo Chu**[3] · **Chun-Fa Chang**[4]

Abstract Grass swaying simulation with respect to wind force plays an important role in the outdoor scene design of video games and animations. However, the complex and dynamic interactions between grass and wind largely hinder the existing approaches from generating physically plausible simulation in real-time performance. Therefore, common approaches compromise by either rendering still meadow or simply adopting a procedural method for simulating the grass motion. In this work, we present a simple yet effective grass model that enables the real-time simulation of grass swaying mimicking real-world grass motions under dynamic wind force. We characterize each individual grass using a simple polyline model with four vertices derived from the control knots of a cubic Bezier curve describing the real grass shape. The grass dynamics is modeled by applying a combination of swinging, bending and twisting motions to the polyline model in response to the input wind force. The deformed grass model is then passed to the shader pipeline to synthesize grass blades for the rendering. Experimental results show that our system not only achieves real-time performance in simulation and rendering, but also scales well to large grass field such as a meadow.

Keywords Grass swaying · Real time · Simulation · Grid-based fluid dynamics

1 Introduction

Grass, lawn, and meadow are common features of outdoor scenes. However, animating the grass motion under the influence of wind could be a daunting task especially if the subtle variations between grass blades are considered. Collectively, those variations of individual grass blades may produce interesting phenomena, such as the eye-catching wave-like motion of a meadow. With the ubiquity of powerful graphics processors (GPUs), we believe the time has come for us to explore the possibility to model and simulate the shape and motion of individual grass blade and see what phenomena are made possible when those grass blades form a meadow collectively.

It is known that hair or fur simulation is commonly adopted by artists when dealing with grass modeling and simulation. However, grass is a completely different type of object compared to the hair or fur. The simulation of hair with respect to wind force is not exactly the same as grass simulation. Artists have to spend a lot of time adjusting parameters in hair simulation just to obtain a reasonable response that is close to grass swaying under different wind conditions. It is even more difficult when several wind forces are acting on a meadow which consists of a large amount of grass. Therefore, a real-time grass simulation system is required to assist the artists in their works.

The simulation of real-world grass–wind interaction involves sophisticated modelings of wind and grass dynamics. For the wind modeling, common fluid dynamics can be applied. However, the grass blades are considered as obstacles which will alter the wind velocity and direction at

Electronic supplementary material The online version of this article (doi:10.1007/s00371-016-1263-7) contains supplementary material, which is available to authorized users.

✉ Ruen-Rone Lee
rrlee@itri.org.tw

[1] Industrial Technology Research Institute, Hsinchu, Taiwan

[2] MediaTek Inc., Hsinchu, Taiwan

[3] National Tsing-Hua University, Hsinchu, Taiwan

[4] National Taiwan Normal University, Taipei, Taiwan

Published online: 10 May 2016

Fig. 1 Snapshots of a small bunch of grass swaying in real world

any instance. Besides, the grass blades change their shapes, height, and orientation due to grass motion with respect to wind force, making wind modeling a very complex task. Grass dynamics requires modeling the characteristics of grass motion under the wind field. It is difficult to work out an integrated physical formulation that models all the grass dynamics in real world. Instead, in order to derive a real-time simulation of grass dynamics, we have to simplify both the wind and the grass modeling. First, we consider only the response from the wind to the grass, but ignore the response from the grass to the wind in order to reduce the complexity of wind modeling. Second, by observing the grass motions under the wind force in real world, we propose a simple grass model and simulate the grass dynamics by a combination of three major grass motions. Besides, the collision detection is also neglected. With these simplifications, we are able to simulate the grass–wind interaction for each individual grass blade of a large grass field in real time.

In this work, we have developed a framework that simulates the grass dynamics under a plausible wind field. We first test it with a small bunch of grass blades and validate it with the motion of real-world grass as in Fig. 1. By utilizing the tessellation shaders and the geometry shaders of modern GPUs, we make our grass model as efficient as possible so that it scales well to a large meadow, which is also exposed to a plausible wind field simulation. As shown in our accompanying video, our grass simulation runs in real time for meadow scenes consisted of hundreds of thousands of grass blades, and produces convincing wave-like motions when the wind blows over the meadow.

Our major contributions can be summarized as follows:

- A framework to simulate the grass swaying under a physically plausible wind field is introduced. The grass–wind interaction can be well simulated with results comparable to real-world grass dynamics.
- An efficient grass model with four knots to control the grass shape is presented and a novel approach to solve the grass dynamics with a combination of swinging, bending, and twisting, under a given wind field is also presented.
- The actual grass geometry is synthesized at run time with a dynamic level-of-detail method and the process can be completely accelerated by a graphics hardware through shader pipeline.

2 Related works

In the past, a high-quality grass simulation had been considered too complex for real-time applications because of the quantities and complexity of physical interaction. Early approach modeled grass through a stochastic process, instancing the actual blades for rendering in buckets and animating in a procedural method [16,21,29]. However, their approaches did not adopt the physical-based models. More specifically, they did not consider the factors, such as wind direction, wind speed, and grass blade orientation, that might cause different grass motions due to grass–wind interaction. Other approaches simplified this process by using billboard [15]. But, it looks unrealistic from some viewing directions.

There were research works dealing with grass–object interaction and using GPU or certain acceleration techniques to speed up the simulation process [3,7,12,20]. The main focus is to render the grass behavior when an object collides with the grass. The results either lack a physical model or produce an unrealistic motion visually. While their works can animate the grass with dynamic objects, our works focus on the effects of grass–wind interaction. We adopt some physical models to simulate the grass motion with respect to the given dynamic wind sources and the results are efficient and visually realistic.

Physical wind modeling is commonly based on computational fluid dynamics. The two most common used techniques are Eulerian grid-based methods [6,8,10] and smoothed particle hydrodynamics (SPH) [9,14]. Early works in the field of Eulerian grid-based fluid dynamics discretize the simulation domain into computational elements. In order to speed up the simulation, Stam introduced the semi-Lagrangian method for advection [24] and presented an efficient implementation of a fluid dynamics solver [25]. Later, Harris presented an implementation which utilize the GPU capability [11]. There were also researches which used particle-based fluid dynamics (SPH) to simulate the inter-

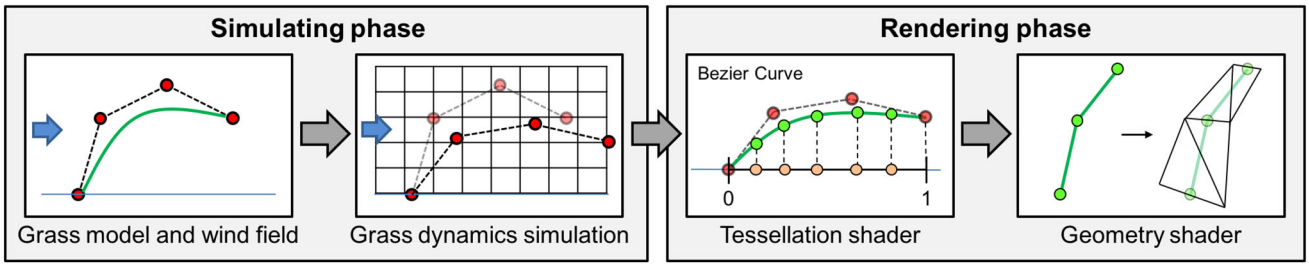


Fig. 2 System overview: inputs to the simulating phase are a grass model characterized as a polyline with four vertices derived from a cubic Bezier curve, and a configuration of wind field (e.g., position, direction, strength, etc.). The system adopts a grid-based fluid dynamics to simulate the physically plausible wind force. When the wind acts on the grass, the system computes the grass dynamics by integrating three common grass motions in real world, namely swinging, bending and twisting, which are caused by external wind force as well as the intrinsic restoration force of grass blade. The estimated motion is then applied to the polyline to obtain the deformed grass shape. In the rendering phase, the system first fits the deformed polyline vertices with a cubic Bezier curve, which is further approximated using line segments in tessellation shader. Lastly, the system draws on each line segment a quadrilateral blade in the geometry shader to render the final grass

action between wind and trees [1,17,22]. Specifically, the botanical tree models [17,18] can generate realistic trees and interact with wind naturally with wind particles, but the performance drops when the number of trees and the number of wind particles increased such as multiple wind sources. The trees and grass are different in structure and thus require a different model to simulate the behavior of grass motion under wind forces. FEM-based simulation is another method to solve the interaction between wind and plants [13,30]. Among those different approaches of wind modeling mentioned above, for performance consideration, we adopt the Eulerian grid-based approach to better fit in our grass dynamics simulation with proposed grass model in this paper.

Coupling wind models with grass are important aspects for grass simulation. Stam [23] modeled the motion between flexible structure and turbulence as a random process. Bakay et al. [2] rendered grass through Russian-doll style transparent shells in the vertex shader and the animation was implemented by moving each vertex along the wind vector which was computed in a preprocessing step. Other grass modelings such as hair, rod, and strand models [4,5,27] are also plausible, but their methods are similar to fur simulation which cannot correctly express the real-world grass characteristic such as swaying and blade twisting with respect to wind forces. There were also some works which adopted physical-based models to calculate the grass motion by dynamically deforming its skeleton according to a predefined wind vector field [3,19,26]. Wang et al. [26] further applied the LOD technique which adapted to different modeling resolutions of a blade to accelerate the rendering process. However, they did not integrate a physically plausible wind model and the computation for each grass blade is too heavy to run in real time for large scenes. In contrast to their method, we simulate the grass dynamics in a combination of different grass motions which were derived from a real-world grass swaying video and can be rendered efficiently through the GPU shader pipeline.

3 Overview

Simulating grass swaying with dynamic wind force involves two phases (see Fig. 2). In the simulating phase, the system takes as inputs a grass represented as a cubic Bezier curve and a wind field configuration describing the position, direction and strength of wind sources. In order to achieve real-time simulation, we propose a simple yet effective grass model to characterize the shape of grass using a parametric curve with four control vertices and form a polyline with those four vertices (Sect. 4). The system first simulates the wind forces by solving a typical Eulerian grid-based fluid model (Sect. 5.1). Then the grass dynamics is solved by integrating three common grass motions in real world, including swinging, bending, and twisting, which are caused by external wind force as well as the intrinsic restoration force of grass blade. The estimated motion is applied to the polyline vertices to obtain the deformed grass shape (Sect. 5.2). In the rendering phase, the system exploits the capability of modern graphics hardware to render the grass geometry based on the coarse polyline model (Sect. 6). Specifically, in the tessellation shader, the system generates a cubic Bezier curve based on the deformed polyline vertices and approximates the curve using a set of line segments. The final grass geometry is rendered in the geometry shader by drawing a quadrilateral on each line segment to represent the grass blade. We also employ a simple level-of-detail technique in the tessellation shader to control the density of line segments according to the viewing distance and to further improve the performance in rendering.

4 Grass modeling

Real-world grass generally appears in the form of flat, smooth and curved shape, and can be easily modeled using a quadrilateral strip to approximate the geometry of grass blade.

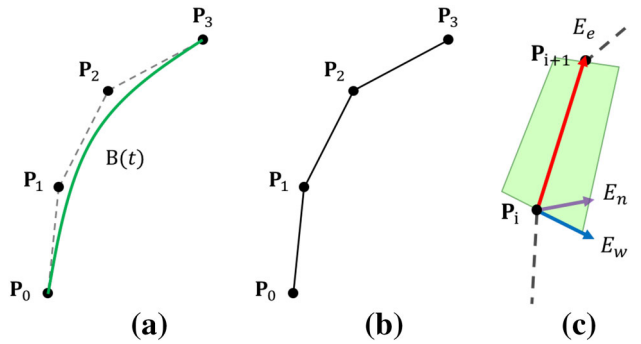


Fig. 3 Representation of grass model. **a** A grass shape is determined by a cubic Bezier curve. **b** The grass model is characterized using a polyline with four vertices derived from control knots in (**a**). **c** Three local vectors are used to define the orientation of grass blade

However, the grass dynamics simulation typically involves computing the motion of hundreds of thousands of grass blades (e.g., a meadow). Thus, direct simulation on individual grass blade with moderate number of vertices is infeasible for real-time applications (e.g., video games) due to intensive computational burden in both physical simulation and rendering. Existing works usually make a trade-off between the physical plausibility of grass motion and the visual quality of grass rendering. For the purpose of generating plausible grass simulation mimicking real-world grass swaying in dynamic wind forces, our key insight is to characterize the grass using a simple model such that the computation of grass dynamics involves only a few geometric primitives (e.g., vertices, edges), while leveraging the capability of powerful graphics hardware (GPUs) for rendering high-quality grass geometry.

The smooth and curved natures of grass blade in real world intuitively lead to a representation using parametric curve. We found that the cubic Bezier curve with one inflection point can express most kinds of grass shape well (see Fig. 3a). Therefore, we use the control vertices of Bezier curve to characterize the grass model and form a polyline with those vertices (see Fig. 3b). We denote the grass model as $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$ are four vertices derived from the control knots of cubic Bezier curve defined as follows:

$$B(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3, \quad 0 \le t \le 1, \tag{1}$$

where t is the interpolation parameter from grass root to tip. The edge set $\mathbf{E} = \{\overline{\mathbf{P}_i\mathbf{P}_j} \mid 0 \le i \le 2, j = i+1\}$ defines a hierarchical linkage structure of grass model. To model grass motion as well as shape of grass blade, we introduce three vectors E_e, E_w, and E_n to represent the blade orientation (see Fig. 3c). E_e is the vector from $\mathbf{P}_i$ to $\mathbf{P}_j$; E_w is a normalized vector orthogonal to E_e; and E_n is the normalized normal vector to the blade segment. A grass blade could be in either static or current state, which indicates, respectively, the grass blade in initial static state or in a specific time step as the wind force is activated. For clarity, we will use a superscript to indicate the state explicitly. For example, $E_\mathrm{n}^\mathrm{static}$ indicates the blade normal vector in the static state.

5 Simulating phase

5.1 Wind field simulation

A physically plausible wind field is commonly described by the Navier–Stokes equation, which can be solved using computational fluid dynamics system [6]. Thus, we simulate fluids by solving the inviscid Navier–Stokes equation,

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{\mathbf{f}}{\rho} - \frac{\nabla p}{\rho}, \tag{2}$$

subject to the incompressibility constraint

$$\nabla\cdot\mathbf{u} = 0, \tag{3}$$

where $\mathbf{u}$ is the fluid velocity field, p is the pressure, t is time, ρ is the fluid density and $\mathbf{f}$ is a field of external forces. Among all possible solutions, Eulerian grid-based approaches possess several advantageous properties over particle-based alternatives such as with higher numerical accuracy, being scalable to support versatile wind field designs (e.g., multiple wind sources, turbulence effect, obstacles), and can be solved directly on GPU. Therefore, we employ a Eulerian grid-based fluid dynamics model to solve the Eq. 2. First, the system discretizes the space of simulation into volumetric grid cells [11], and we use the grid resolution of $60 \times 60 \times 20$ in our implementation. When the wind field is activated, the system adds the external forces to the grid cells where the wind sources are located in. Then, the velocity field is advected based on a semi-Lagrangian advection scheme [24], followed by solving the pressure using Jacobi iteration method [28] to ensure the incompressibility constraint.

5.2 Grass dynamics simulation

When the wind acts on a grass, its shape deforms according to direction and speed of wind force. For instance, the grass waggles back-and-forth around its static configuration in the gentle breeze, while a large-scale swaying effect is observed under the strong wind (see supplementary video). To capture various grass motions under dynamic wind force, we propose a novel formulation that models the grass dynamics as a combination of three common grass motion patterns, namely swinging, bending, and twisting, acting on the poly-

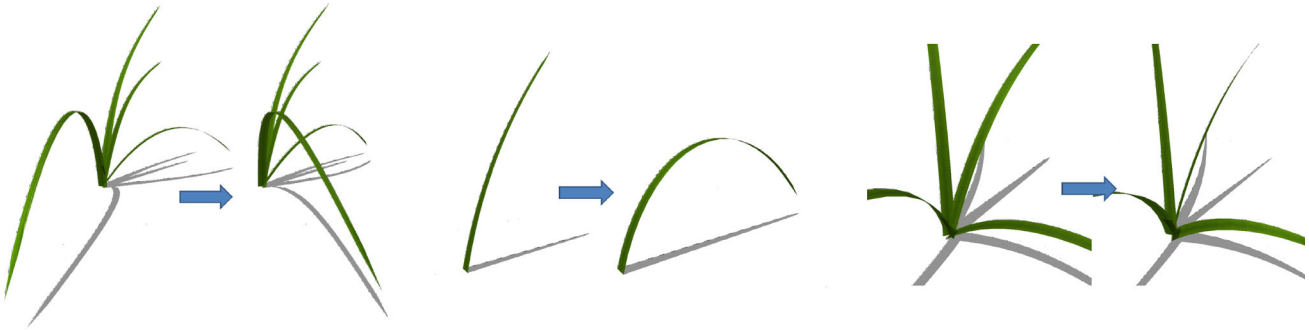


Fig. 4 Three common grass motions in real world. *Left* grass swinging. *Middle* grass bending. *Right* grass twisting

line grass model. Swinging represents a large-scale motion where the whole grass blade is dragged by the wind (see Fig. 4, left). Grass bending depicts the degree of grass blade bending along the nervure direction (see Fig. 4, middle). Grass twisting illustrates the changes in the orientation of grass blades (see Fig. 4, right). In addition to external wind force, the real-world grass motion also shows an oscillating pattern in which the grass tends to restore back to its original shape as the wind stops. We model such intrinsic restoration force as a damped harmonic motion.

External wind force. According to aerodynamic drag equation, we formulate the external wind force as:

$$W = S\sigma\,\overrightarrow{v}, \tag{4}$$

where S is the area of the thrust surface of wind, σ is the drag coefficient and $\overrightarrow{v}$ represents the wind velocity.

Intrinsic restoration force. There exists an intrinsic restoration force to move a deflected (i.e., deformed) grass blade toward its static state. Its strength depends on the grass stiffness and is proportional to the magnitude of grass motion. Thus we define the restoration force as:

$$R = k\Delta\theta\,\Delta\overrightarrow{s}, \tag{5}$$

where k is the stiffness coefficient, $\Delta\theta$ and $\Delta\overrightarrow{s}$ represent, respectively, the angular displacement and the normalized vector from current position to static position that capture the grass motion.

Damping force. To model the oscillating motion of grass around its static state, we introduce a damping force with amplitude decreasing over time when the wind stops, and is defined as:

$$D = -c\,\overrightarrow{\omega}, \tag{6}$$

where c and $\overrightarrow{\omega}$ represent the grass damping coefficient and angular velocity, respectively.

The total force that acts on the grass model is thus expressed as:

$$F = W + R + D. \tag{7}$$

Since the grass model is represented as hierarchical linked edge structure, the grass dynamics can be solved by computing the forces and estimating the corresponding motion on each edge. To this end, we first compute the torque N as the cross product of the edge vector E_e and the total force F: $N = E_e \times F$. Then, for each edge, the derivative of the angular velocity $\overrightarrow{\omega}$ is estimated using the torque N and the moment of inertia $\mathbf{I}$.

$$N = \mathbf{I}\overrightarrow{\alpha} = \mathbf{I}\frac{d\overrightarrow{\omega}}{dt}; \quad \mathbf{I} = mr^2, \tag{8}$$

where m and r are the mass and length of edge E_e, respectively. Finally, for each frame, we use the explicit Euler time integration to solve the angular displacement $\Delta\theta$.

$$\begin{aligned} \overrightarrow{\omega}' &= \overrightarrow{\omega} + \overrightarrow{\alpha}\,\Delta t \\ \Delta\theta &= \overrightarrow{\omega}'\Delta t. \end{aligned} \tag{9}$$

However, due to the piecewise polyline representation of grass model, applying above force dynamics to individual edge may cause inconsistent motions between adjacent edges. For instance, at some moment of simulation, some edges might be dragged by the wind force while some are restoring to their static state. Figure 5 shows some examples of inconsistent motion, where the arrows represent wind force directions or restoration force directions. Such inconsistency will numerically lead to intractable simulation process. To tackle such problem, we sample the wind velocity only at the tip of grass and propagate the estimated grass motion toward the grass root to obtain a smooth grass deformation. In the following paragraphs, we will elaborate how to calculate the forces according to the reaction of grass to wind force in terms of swinging, bending and twisting.

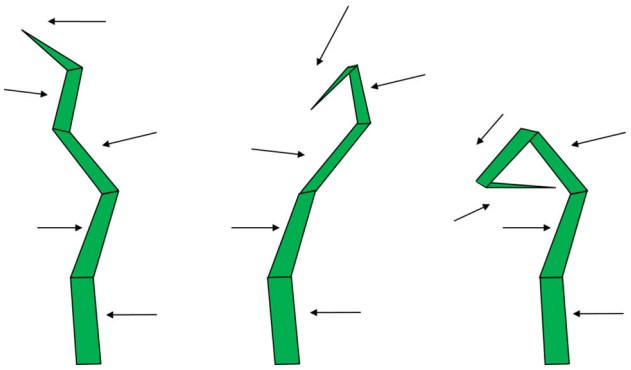


Fig. 5 Illustrations of inconsistent motions between adjacent edges

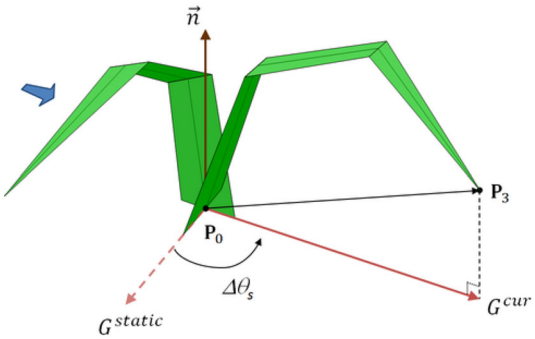


Fig. 6 The grass swinging model

Swinging. Figure 6 illustrates the swinging motion under a wind force. We can see that the grass swinging behaves like the whole grass rotating about a certain axis. To model such motion, we first define a growth vector G to represent the growth direction of each grass by projecting $\overrightarrow{\mathbf{P}_0\mathbf{P}_3}$ to the ground plane. Then, the grass swinging motion can be modeled as a rotation of growth vector relative to the normal of the ground plane. To compute the wind force, we project the wind velocity $\vec{v}$ to the vector E_w, and define the force as:

$$W_\mathrm{S} = S\sigma(\vec{v} \cdot E_\mathrm{w})E_\mathrm{w}. \tag{10}$$

Similarly, we use the growth vector G to define the restoration force as:

$$R_\mathrm{S} = k_\mathrm{tip}\Delta\theta_\mathrm{s}\frac{G^\mathrm{static} - G^\mathrm{cur}}{|G^\mathrm{static} - G^\mathrm{cur}|}, \tag{11}$$

where k_tip is the stiffness coefficient of the tip edge in grass model and $\Delta\theta_\mathrm{s}$ is the swinging angular displacement. We further assume that the edge close to grass root is more rigid than the one at grass tip and increase the stiffness from grass tip to root. The swinging angular displacement $\Delta\theta_{i,\mathrm{s}}$ for the ith edge is then defined as:

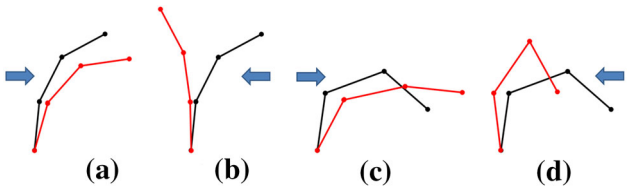


Fig. 7 Illustrating four types of grass bending effects. The wind force is represented by the *blue arrow*. **a**, **b** The bending directions are consistent across edges. **c**, **d** The tip and root edges bend in opposite directions

$$\Delta\theta_{i,\mathrm{s}} = \frac{k_\mathrm{tip}}{k_i}\Delta\theta_\mathrm{s}, \tag{12}$$

where k_i represents the stiffness coefficient of the ith edge. This is the way we propagate the motion from the tip to the root. The method ensures a consistent and smooth swinging motion among all the edges in the grass model.

Bending. The shape of grass may also bend along the nervure direction under the wind force. Since the bending motion is only driven by the projection of wind force to the blade normal vector of each edge, we define the wind force as:

$$W_\mathrm{B} = S\sigma(\vec{v} \cdot E_\mathrm{n})E_\mathrm{n}. \tag{13}$$

Similar to Eq. 11, the restoration force is defined as:

$$R_\mathrm{B} = k_\mathrm{tip}\Delta\theta_\mathrm{b}\frac{E_\mathrm{e}^\mathrm{static} - E_\mathrm{e}^\mathrm{cur}}{|E_\mathrm{e}^\mathrm{static} - E_\mathrm{e}^\mathrm{cur}|}, \tag{14}$$

where $\Delta\theta_\mathrm{b}$ is the bending angular displacement. However, for some specific shape of grass such as the one with drooping shape, the edges may bend in opposite directions in a grass model. For example, Fig. 7a, b illustrate the cases that the bending directions are consistent across all edges while Fig. 7c, d are not. Therefore, a direct propagation of the bending transformation on the tip edge may result in weird and unrealistic grass shape. To tackle this problem, we first split the edges of a grass model into two groups based on the sign of edge slope. Then, we compute force on a local tip edge in each group and apply the estimated motion to remaining edges in the same group. Likewise, similar to Eq. 12, we can also define the bending angular displacement $\Delta\theta_{i,\mathrm{b}}$ for the ith edge to propagate the bending motion from the local grass tip to the local grass root.

Twisting. The twisting motion is modeled as the rotation of each grass blade about its edge vector E_e (see Fig. 8a). However, twisting a blade will change its normal vector E_n, which determines the direction of bending, and vice versa. Such mutual correlation will render unpredictable grass motion. To solve this problem, we model the blade twisting as a local transformation that does not alter the bending direction (Fig. 8c). The key step is to copy the vector E_w to a local

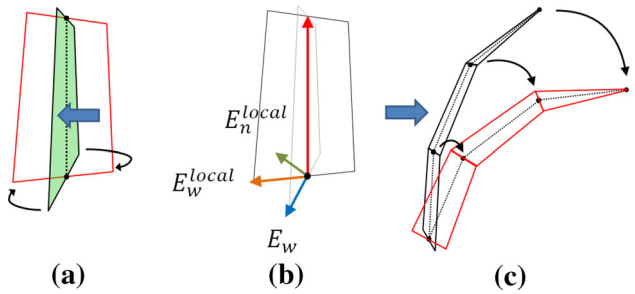


Fig. 8 The grass blade twisting model

vector $E_{\mathrm{w}}^{\mathrm{local}}$ in the beginning and treat $E_{\mathrm{w}}^{\mathrm{cur}}$ as a local vector in static state. Then, all the computations are carried out on $E_{\mathrm{w}}^{\mathrm{local}}$ instead of E_{w} (see Fig. 8b). Hence, the wind force is defined as:

$$W_{\mathrm{T}} = S\sigma\left(\overrightarrow{v} \cdot E_{\mathrm{n}}^{\mathrm{local}}\right) E_{\mathrm{n}}^{\mathrm{local}}, \tag{15}$$

where $E_{\mathrm{n}}^{\mathrm{local}} = E_{\mathrm{w}}^{\mathrm{local}} \times E_{\mathrm{e}}^{\mathrm{cur}}$ is a local normal vector. And the restoration force is calculated using $E_{\mathrm{w}}^{\mathrm{cur}}$ and $E_{\mathrm{w}}^{\mathrm{local}}$ and defined as:

$$R_{\mathrm{T}} = k_{\mathrm{tip}}\Delta\theta_{\mathrm{t}}\frac{E_{\mathrm{w}}^{\mathrm{cur}} - E_{\mathrm{w}}^{\mathrm{local}}}{|E_{\mathrm{w}}^{\mathrm{cur}} - E_{\mathrm{w}}^{\mathrm{local}}|}, \tag{16}$$

where $\Delta\theta_{\mathrm{t}}$ is the twisting angular displacement. Again, similar to Eq. 12, we can also define the twisting angular displacement $\Delta\theta_{i,\mathrm{t}}$ for the ith edge to propagate the twisting motion from the grass tip to the grass root. Finally, we apply the transformation to $E_{\mathrm{w}}^{\mathrm{local}}$ for each edge and use the result as final grass blade twisting model for rendering.

6 Rendering phase

The render phase is to restore back the grass geometry from the simplified grass model and synthesize the grass blade for rendering. We implement this process through a GPU shader pipeline (see Fig. 9). The grass model (static or deformed) is stored in the vertex buffer and read into the vertex shader first. The four vertices are treated as the control points of a cubic Bezier curve segment, based on which the Tessellation Control Shader determines the tessellation level of detail. Specifically, the tessellation level represents the number of samples needed to be generated for the curve segment. The higher the tessellation level, the smoother the grass shape. However, it will slow down the rendering process for large scenes. Thus, we adopt the concept of level of detail (LOD) which tessellates the grass according to the distance to the viewer. In our implementation, the tessellation levels have been set to 20, 10, and 3 for close, medium, and far distances to the viewer, respectively. Once the tessellation level is determined. The Primitive Generator (PG) is used to generate the sample points, in parametric form ranging from 0 to 1, on the Bezier curve. Then, the vertices and the associated orientation vectors, as well as texture coordinates, on the actual grass curve segment can be derived by the Tessellation Evaluation Shader.

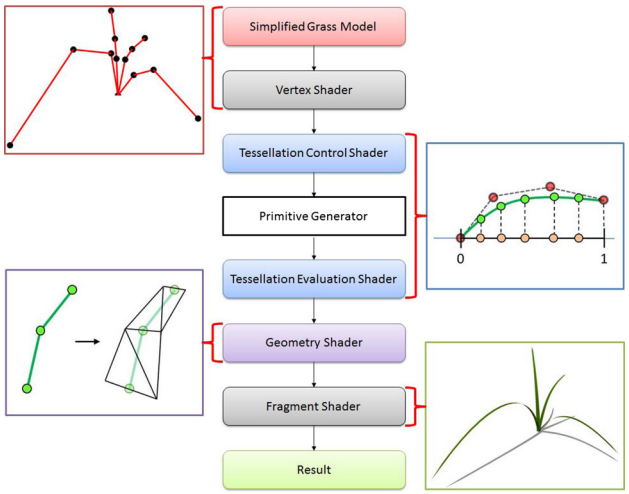


Fig. 9 Shader pipeline for grass blade synthesis process

The actual grass blade is generated by the Geometry Shader (GS). By expanding each line segment, within the curve segment, into a quadrilateral with given orientation vector, a triangle strip with four vertices which defined a section of grass blade is generated. Finally, each blade section is rendered by the Fragment Shader (FS) with grass texture applied. The grass blade synthesis is efficient, because all the derivation can be realized by the shader pipeline.

7 Results

We implemented our system in C++ and OpenGL on a desktop PC with an Intel Xeon E3-1230 CPU running at 3.3 GHz, 16 GB of memory installed, and an NVIDIA Geforce 670 GTX for graphics hardware acceleration. We employed the tessellation and geometry processing capabilities of GLSL 4.0 to synthesize and render the grass blade including two-side lighting and shadows with a shadow map of resolution 8192 × 8192. All the results are rendered in 8× MSAA. Table 1 summarizes the performance of our framework. With LOD, our simulation can achieve real-time performance for 100 K grass blades and in interactive time for 150 K grass blades.

Figure 10 shows some simulation results with wind force coming from the left side for a few bunches and several bunches of grass. A snapshot of simulating a meadow-scale grass swaying under a rotating wind force can be found in Fig. 11. Figure 12 illustrates the grass rendered with differ-

Table 1 Simulation performance

no. of Grasses	Wind Sim (ms)	Grass Sim (ms)	Grass rendering (ms)		FPS	
			w/o LOD	w. LOD	w/o LOD	w. LOD
50K	4.2	8.5	10.3	8.3	43.5	47.5
100K	4.2	16.9	15.8	12.1	27.1	30.1
150K	4.2	25.8	20.7	15.0	19.7	22.2
200K	4.2	33.5	25.2	18.1	15.9	17.9

Fig. 10 *Left column* a few (*top*) and several (*bottom*) bunches of static grass. *Right column* simulating grass swaying in a wind field using our system

Fig. 11 Simulating a meadow-scale grass swaying under a dynamic wind force shown on the *right*

Fig. 12 Level-of-detail on grass

ent level of detail. An extension of our framework to support wind force being blocked by some obstacles in the scene is demonstrated in Fig. 13. The grass swaying simulation is best demonstrated in animation. Please refer to the supplementary video for all the effects including swinging, bending, and twisting for small, medium, and large bunches of grass animation.

Fig. 13 Effect of grass swaying with obstacles

8 Conclusions and future works

In this paper, we present a grass swaying framework with dynamic wind force. Each grass blade is represented by a coarse grass model to control its shape. The grass model is used in grass dynamics simulation and in later synthesizing the actual grass blade. A novel approach is introduced to solve the grass dynamics with a combination of swinging, shape bending, and blade twisting under a wind field simulated by Eulerian grid-based fluid dynamics. All the grass blade synthesis and rendering are realized by the hardware shader pipeline. To further accelerate the rendering process, we also adopt the technique of level-of-detail that dynamically determines the number of blade segments. By simplifying the detailed grass blade to a coarse one, we can greatly reduce the computations with wind forces and also derive convincing results in comparison to the real-world grass swaying motion.

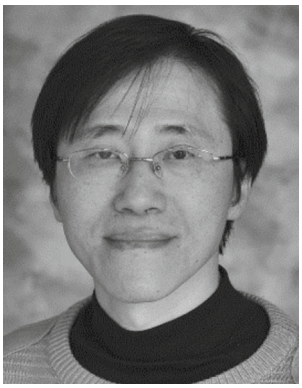
Although our framework can simulate the grass swaying under a wind field, there are still works we would like to do in the future. For example, in our simulation, we only consider the wind effect acting on grass, but ignore the response of wind bouncing back from the grass. Considering a grass with wider blade, this factor will become more essential for the grass dynamics simulation. Moreover, we did not consider the collision behavior between grass blades. Detecting collision between grass can apply common bounding volume to achieve it. However, the grass response after detecting collision is much more complicated then grass swaying. We will deal with these issues in our future works.

Acknowledgments We are grateful to the anonymous reviewers for their comments, suggestions, and additional references. The project

was supported in part by the Ministry of Science and Technology (MOST-102-2221-E-007-055-MY3 and MOST-103-2221-E-007-065-MY3) and the Ministry of Economic Affairs (MOEA-105-EC-17-A-24-1177), Taiwan.

References

1. Akagi, Y., Kitajima, K.: A study on the animations of swaying and breaking trees based on a particle-based simulation. J. WSCG **20**(1), 21–28 (2012)
2. Bakay, B., Lalonde, P., Heidrich, W.: Real-time animated grass. In: Eurographics 2002. Eurographics (2002)
3. Belyaev, S., Laevsky, I., Chukanov, V.: Real-time animation, collision and rendering of grassland. In: Proceedings of GraphiCon2011, pp. 1–4 (2011)
4. Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., Grinspun, E.: Discrete elastic rods. ACM Trans. Graph. (TOG) **27**(3), 63:1–63:12 (2008)
5. Bertails, F.: Linear time super-helices. Comput. Graph. Forum **28**(2), 417–426 (2009)
6. Bridson, R.: Fluid Simulation for Computer Graphics. CRC Press, Boca Raton (2008)
7. Fan, Z., Li, H., Hillesland, K., Sheng, B.: Simulation and rendering for millions of grass blades. In: Proceedings of the 19th symposium on interactive 3D graphics and games, pp. 55–60. ACM (2015)
8. Foster, N., Metaxas, D.: Realistic animation of liquids. Graph. Models Image Process. **58**(5), 471–483 (1996)
9. Gingold, R.A., Monaghan, J.J.: Smoothed particle hydrodynamics: theory and application to non-spherical stars. Mon. Not. R. Astron. Soc. **181**(3), 375–389 (1977)
10. Harlow, F.H., Welch, J.E., et al.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids **8**(12), 2182–2189 (1965)
11. Harris, M.: Fast fluid dynamics simulation on the gpu. GPU Gems **1**, 637–665 (2004)
12. Jens, O., Salama, C.R., Kolb, A.: Gpu-based responsive grass. J. WSCG **17**(1–3), 65–72 (2009)
13. Lu, S., Guo, X., Zhao, C., Li, C.: Physical model for interactive deformation of 3d plant. Int. J. Virtual Real. **10**(2), 33–38 (2011)
14. Lucy, L.B.: A numerical approach to the testing of the fission hypothesis. Astron. J. **82**, 1013–1024 (1977)
15. Pelzer, K.: Rendering countless blades of waving grass. GPU Gems **1**, 107–121 (2004)
16. Perbet, F., Cani, M.P.: Animating prairies in real-time. In: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 103–110. ACM (2001)
17. Pirk, S., Niese, T., Hädrich, T., Benes, B., Deussen, O.: Windy trees: computing stress response for developmental tree models. ACM Trans. Graph. **33**(6), 204:1–204:11 (2014)
18. Pirk, S., Stava, O., Kratt, J., Massih Said, M.A., Neubert, B., Mech, R., Benes, B., Deussen, O.: Plastic trees: interactive self-adapting botanical tree models. ACM Trans. Graph. **31**(4), 50:1–50:10 (2012)
19. Qiu, H., Chen, L., Chen, J.X., Liu, Y.: Dynamic simulation of grass field swaying in wind. J. Softw. **7**(2), 431–439 (2012)
20. Qiu, H., Chen, L.T., Qiu, G.P.: A novel approach to simulate the interaction between grass and dynamic objects. WSEAS Trans. Comput. **12**(7), 277–287 (2013)
21. Reeves, W.T., Blau, R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. ACM Siggraph Comput. Graph. **19**(3), 313–322 (1985)
22. Selino, A., Jones, M.: Large and small eddies matter: animating trees in wind using coarse fluid simulation and synthetic turbulence. Comput. Graph. Forum **32**(1), 75–84 (2013)
23. Stam, J.: Stochastic dynamics: Simulating the effects of turbulence on flexible structures. Comput. Graph. Forum **16**(s3), C159–C164 (1997)
24. Stam, J.: Stable fluids. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques, pp. 121–128. ACM Press (1999)
25. Stam, J.: Real-time fluid dynamics for games. In: Proceedings of the game developer conference, vol. 4, pp. 76–92. UBM Tech (2003)
26. Wang, C., Wang, Z., Zhou, Q., Song, C., Guan, Y., Peng, Q.: Dynamic modeling and rendering of grass wagging in wind. Comput. Anim. Virtual Worlds **16**(3–4), 377–389 (2005)
27. Ward, K., Bertails, F., Kim, T.Y., Marschner, S.R., Cani, M.P., Lin, M.C.: A survey on hair modeling: styling, simulation, and rendering. IEEE Trans. Visualiz. Comput. Graph. **13**(2), 213–234 (2007)
28. Wilkinson, J.H.: The algebraic eigenvalue problem, vol. 87. Clarendon Press, Oxford (1965)
29. Zhao, X., Li, F., Zhan, S.: Real-time animating and rendering of large scale grass scenery on gpu. In: International conference on information technology and computer science, ITCS 2009, vol. 1, pp. 601–604. IEEE (2009)
30. Zhao, Y., Barbič, J.: Interactive authoring of simulation-ready plants. ACM Trans. Graph. (TOG) **32**(4), 84:1–84:12 (2013)



Ruen-Rone Lee received his Ph.D. degree in computer science from National Tsing Hua University, Taiwan, in 1994. From 1994 to 2010, he worked in companies for graphics hardware and software development. Later, he joined the Department of Computer Science, National Tsing Hua University, as an associate researcher from 2010 to 2015. He is currently a deputy technical director in the Information and Communications Research Laboratories, Industrial Technology Research Institute, Taiwan. His research interests include computer graphics, non-photorealistic rendering, and graphics hardware architecture design.

Yi Lo received the M.S. degree from the Department of Computer Science, Nation Tsing Hua University, Taiwan, in 2015. His research interests focus on real-time physics simulation and rendering. Currently, he is a software engineer at MediaTek Inc.

Hung-Kuo Chu is an assistant professor at the Department of Computer Science, National Tsing Hua University. He received B.S. and Ph.D. degrees from Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests focus on Shape Understanding, Smart Manipulation, Perception-based Rendering, Recreational Graphics and Human Computer Interaction.

Chun-Fa Chang received the B.S. degree from National Taiwan University, the M.S. from the University of Texas at Austin, and the Ph.D. from the University of North Carolina at Chapel Hill. He was a software engineer at Intel Corporation from 1992 to 1995, and a summer intern at DEC-Western Research Lab in 1996 and 1997. He is currently a professor in the Department of Computer Science, National Taiwan Normal University. His research interests include real-time rendering, photo-realistic image synthesis, and applications using multicore processors.