Animating 3D Vegetation in Real-time Using a 2D Approach

Kan Chen[1,2] *　　Henry Johan[2]

[1]Nanyang Technological University　[2]Fraunhofer IDM@NTU

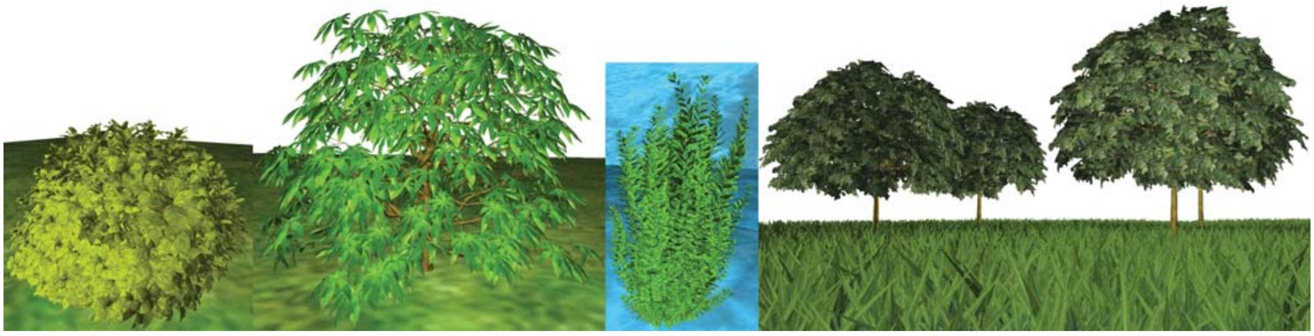


Figure 1: *Our method can be used to animate (from left to right) shrub, tree, underwater vegetation, and vegetation scene.*

Abstract

In this paper, we propose a 2D approach for real-time animation of vegetation in 3D scenes, especially suitable for simulating wind effects on 3D vegetation fields with densely leaved foliage. We represent a vegetation field as view-dependent 2D billboard layers, perform a 2D harmonic motion simulation for modeling the dynamics of vegetation at the first layer (closest to the viewer), and utilize this dynamics to guide the animation of the rest of the layers while addressing the motion effects in depth and occlusion effects. As a result, our method can produce natural looking motions of vegetation swaying in wind comparable with existing commercial software, however the effort to setting up the underlying animation model and the computational cost can be significantly reduced.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: 3D vegetation, real-time animation, 2D billboards, harmonic motion

1 Introduction

Real-time animation of vegetation has many applications, such as in games and simulators. For real-time applications such as games, there are many tasks need to be performed such as AI and character animation, so it is important to lower the computational cost of vegetation animation while still making the animation looks visually plausible. It is a challenging problem, especially for dense vegetation scenes such as shrubs, tree crowns, and grass fields, which usually have complex geometry and may consist of a large quantity of elements (grass blades, leaves, etc.). Most existing 3D animation methods require one stage to convert the "polygon soup" of vegetation to a format suitable for 3D animation by setting up a 3D animation underlying model (bones, joints, etc.). Furthermore, existing 3D animation methods mainly focus on animating single vegetation element in 3D. Because of the huge number of elements, animating a dense vegetation field requires tedious effort to set up the underlying 3D animation model and high computational cost.

In this paper, we focus on realizing the essential visual effects of natural looking motions of a dense vegetation field in real-time. We do not aim to produce physically accurate motions. The targeted visual effects are based on the observations from real scenes:

(1) In dense vegetation, vegetation elements may stick together and move in groups, exhibiting harmonic motion effects such as swaying and oscillating.

(2) In dense vegetation, such as tree crown whose leaves are dense and interlacing, the wind effects occurring at different parts of vegetation may have some similarities.

(3) 3D vegetation may bend towards or away from the viewer and the vegetation's movements may propagate from the back to the front and vice versa. We call this motion effects in depth.

(4) Occlusion effects will occur when leaves or blades overlap the leaves or blades behind. For viewers to perceive the 3D motion of vegetation, it is essential to model motion effects in depth and occlusion effects.

In this paper, we propose an efficient 2D method to animate 3D vegetation with densely leaved foliage under the influence of wind in real-time, which consists of the following features:

• We convert the 3D vegetation "polygon soup" to view-dependent billboard layers (Section 3.2) and perform a 2D simulation of harmonic motion at a simulation plane for modeling the dynamics of vegetation at the first layer (closest to the viewer) (Sections 3.3 and 3.4). Our method does not require tedious operations to set up an underlying 3D animation model.

• We utilize the dynamics of the first layer to animate the rest of the vegetation layers (Section 3.5). We realize (1) the motion effects in depth by considering the interaction between vegetation layers as well as (2) occlusion effects which are generated by the differences in motions between vegetation layers. Our method does not perform an explicit 3D simulation. As a result the computational cost can be reduced, which is favorable for real-time applications.

• Our method is real-time. Users can adjust the results by setting the number of layers in the manner of level-of-detail (LOD) as well as the properties of wind and vegetation to achieve their desired animation effects on the fly.

*e-mail: {kchen1,henryjohan}@ntu.edu.sg

Different from [Chen and Johan 2013], we propose an integrated system for animation of 3D vegetation while handling the change of viewpoint, inter-layer interaction, and motion effects in depth. We only adopt the 2D simulation of vegetation based on harmonic motion from [Chen and Johan 2013].

2 Related work

2.1 Modeling of vegetation

Billboard representation: Billboards are the conventional way to represent grass [Habel et al. 2007; Jensn et al. 2009; Chen and Johan 2010]. Billboard clouds, introduced by [Décoret et al. 2003], represent geometry through a set of arbitrarily oriented billboards. Many methods have been introduced to represent tree models using billboards, including those by [Behrendt et al. 2005], and [Bao et al. 2009]. In general, the billboards for vegetation are determined in advance and do not change on the fly. As a result, this method is efficient in rendering. However much of the original geometry information may be lost due to the simplification, thus it is not suitable for close-up viewing. In contrast to this conventional approach, in our method, we dynamically create view-dependent billboards of vegetation as the viewer moves.
Volumetric representation: [Kajiya and Kay 1989] introduced 3D texture mapping to model a furry surface. [Meyer and Neyret 1998] extended this work to slice 3D geometry into a series of thin layers called volumetric textures. [Lengyel et al. 2001] proposed another extension called the shell based method used mainly for grass. [Decaudin and Neyret 2004] proposed to represent a forest using prisms which are aperiodically mapped onto the terrain. [Decaudin and Neyret 2009] improved this work and introduced volumetric billboards to present plant foliage. Although the volumetric representation can produce realistic results, it makes the editing and animation difficult and requires high memory cost.
3D geometric representation and LOD: The polygon is the primitive to represent the geometric model of a tree. However it is tedious to set up the underlying animation model and requires high computational cost. LOD techniques have been widely used to eliminate minor details and they provide the possibility to render and animate large-scale dynamic forest scenes in real-time. To represent near and far vegetation respectively, [Perbet and Cani 2001] used chains of line-segmented polygons and semi-transparent vertical textures of the same orientation. [Bruneton and Neyret 2012] used 3D models and a texture rendered with view and light dependent shaders.
Particles: [Reeves and Blau 1985] used particles' trajectories to draw grass.

2.2 Animating vegetation in 2D images

Given a single image which consists of vegetation, the following methods can animate the vegetation in it. [Shinya et al. 1999] proposed to animate plants in 2D images by combining physically-based simulation with skeleton-based morphing techniques. [Chuang et al. 2005] proposed to segment an image into layers and animate the image layer by layer with a motion texture. [Chen and Johan 2013] proposed to combine 2D fluid simulation, wave simulation and grid-based image warping to animate vegetation in 2D images.

2.3 Animating trees in 3D scenes

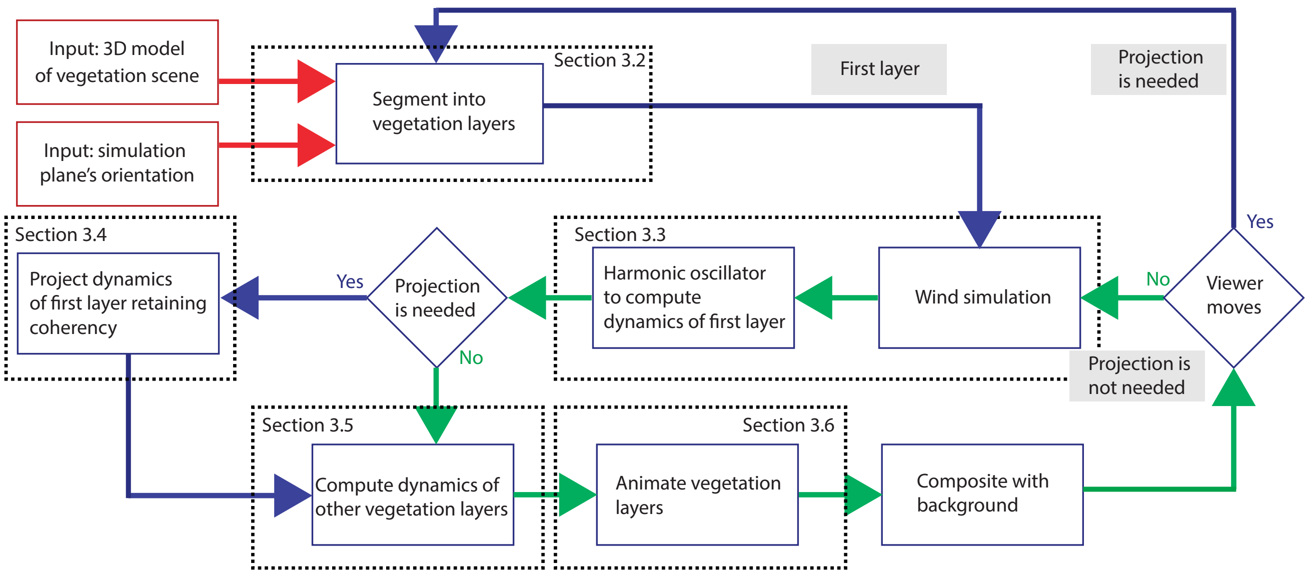
Physics-based methods: These methods are usually based on considering joints as structural nodes and applying explicit or simplified integration of the equations of motion. They in general require high computational cost and lack intuitive direct control. [Stam 1997] introduced modal analysis to animate a tree. The simulation is performed in the frequency domain. To reduce the computational cost, the spectral method is employed. [Chuang et al. 2005] also used a spectral method by computing the motion spectrum of a damped harmonic oscillator to model the animation of plants. [Habel et al. 2009] proposed a physics guided animation using a similar spectral method as in [Chuang et al. 2005] and beam deformation. [Diener et al. 2009] proposed to apply modal analysis to compute a wind basis and project directional wind at run time. A common geometrical model to represent branches is the beam model [Chuang et al. 2005; Habel et al. 2009; Hu et al. 2012]. Moreover, [Weber 2008] presented an interactive simulation method using a cloth dynamics model. [Zhao and Barbič 2013] proposed to convert plant geometry to a format suitable for physically based simulation.
Procedural methods: These methods are usually based on noise functions to heuristically model the appearance of tree motions, for example, in [Ota et al. 2003], [Sousa Hubert Nguyen, 2007] and [Hu et al. 2009; Hu et al. 2012]. Procedural methods are commonly employed to generate the wind effect by using trigonometric functions or noise [Pelzer Randima Femando, 2004; Zioma Hubert Nguyen, 2007]. However [Akagi and Kitajima 2006] performed an explicit fluid simulation to simulate the wind effect. [Ramraj Kim Pallister, 2005] simulated the wind effect using a simple water wave simulation. [SpeedTree] is the state-of-the-art solution for vegetation animation. It has been widely used in real-time applications such as games and simulators as well as movies. The wind effect in [SpeedTree] is achieved based on setting some procedures, such as setting frequency controls how fast branches oscillate. However the technical details are not available in the public domain.
Acceleration methods: These methods play an important role in animating large scale vegetation scenes in real-time. The major methods are: the LOD methods [Beaudoin and Keyser 2004], GPU methods [Habel et al. 2009] and pre-computation and data-driven methods [Zhang et al. 2006; Zhang et al. 2007].

2.4 Animating grass in 3D scenes

[Perbet and Cani 2001] proposed to use a procedural wind model to animate different levels of plant representation. [Guerraz et al. 2003] extended this work to handle vegetation-object interaction. [Wang et al. 2005] used the free form deformation to animate grass blades and the grass-grass collision detections were performed explicitly. [Banisch and Wuthric 2006] performed a mass-spring simulation to animate grass represented using shells. [Habel et al. 2007] proposed to distort the texture lookups to achieve a texture based grass animation. [Jensn et al. 2009] modeled the responsive grass using a spring model. [Chen and Johan 2010] proposed to consider grass field as a continuum and shift the interactions to the dynamics of continuum.

2.5 Summary

For trees, most methods [Shinya et al. 1999] are based on setting up skeletons as the animation model, they are more suitable for tree type vegetation with branches, whose motions are usually rigid. Existing 3D methods [Habel et al. 2009; Hu et al. 2012; Zhao and Barbič 2013] to animate vegetation tend to consider each single element such as leaf, and animate the elements explicitly in 3D. However, for dense vegetation, setting up the underlying animation model and performing an explicit 3D simulation require high computational cost. Since the elements are dense and the elements at the back usually attract less attention and dense vegetation is usually non-rigid, we propose a more efficient way by focusing on the apparent parts and essential effects such as the non-rigid property and the motion effects in depth. To reduce the computational



Input: 3D model of vegetation scene
Input: simulation plane's orientation
Section 3.2
Segment into vegetation layers
First layer
Projection is needed
Section 3.4
Project dynamics of first layer retaining coherency
Yes
Projection is needed
No
Section 3.3
Harmonic oscillator to compute dynamics of first layer
Wind simulation
No
Viewer moves
Yes
Projection is not needed
Section 3.5
Compute dynamics of other vegetation layers
Section 3.6
Animate vegetation layers
Composite with background

Figure 2: *The steps. Red: input. Green: main loop. Blue: update. Dotted boxes: the corresponding sections in the paper.*

cost, instead of directly animating the vegetation in 3D, we perform the animation in 2D space by dynamically generating layers of view-dependent billboards. In contrast to our proposed method, existing 2D methods to animate 2D vegetation images layer by layer [Chuang et al. 2005; Chen and Johan 2013] do not handle change of viewpoint and inter-layer interaction. Thus the motion effects in depth cannot be synthesized. We also compare our animation results with [SpeedTree] in Section 4.

3 Our proposed method

3.1 Basic idea and steps

Basic idea: Given a "polygon soup" of a dense 3D vegetation, we aim to generate the following natural looking motions under the influence of wind in real-time: shrub swaying in wind, tree crown oscillating in breeze, and wave effect of grass field. In real-time applications, the realization of motion effects is more important than the physical accuracy. Hence, we adopt a 2D approach and focus on the "apparent parts" of a vegetation field which are closer to the viewer and usually attract more attention from the viewer. Vegetation motion at different parts may also have similarities. Thus, we propose to represent a 3D vegetation field using view-dependent 2D billboards. To account for wind effects, we set a 2D simulation plane in which we perform a 2D simulation of harmonic motion. We propose to tile and stack the simulation result to model the wind effects in 3D space. The 2D simulation result of harmonic motion is then applied for modeling the vegetation's motion at the first layer closest to the viewer. The resulting dynamics at the first layer is utilized to guide the vegetation's motion in other layers. Since we use billboard layers, we can also easily generate the visual cues (parts of vegetation moving in groups, layer-layer interaction and occlusion effects) for perceiving the 3D motion effects. The motion in dense vegetation field usually exhibit the non-rigid and harmonic property: the vegetation's shape is deforming or oscillating, hence we adopt the method of 2D simulation of harmonic motion from [Chen and Johan 2013], which is a combination of a 2D fluid simulation and a 2D harmonic oscillator. We also extend the 2D grid-based image warping in [Chen and Johan 2013] to simulate the motion

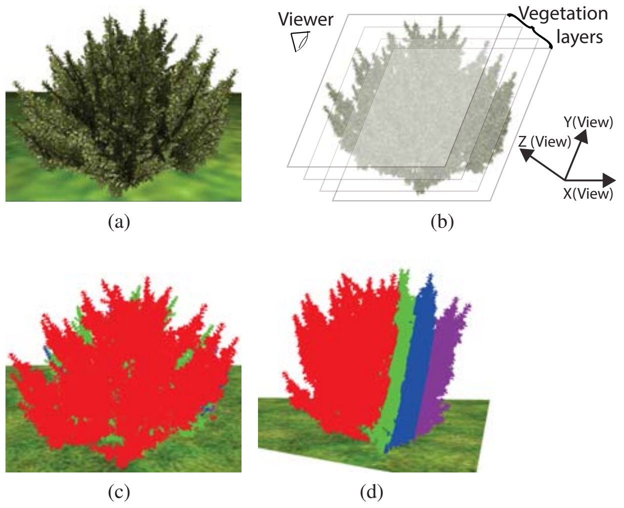


Figure 3: *Representation of a 3D shrub mesh. (a) The input 3D mesh. (b) Segment the mesh into four view-dependent vegetation layers. (c) Four vegetation layers in the viewer's view. Different colors indicate different layers. (d) Side view of (c).*

effects in depth.

Steps: As shown in Figure 2, besides the 3D mesh (Figure 3 (a)), another input for our method is the simulation plane's orientation, it defines the simulation coordinates (Figure 4). After generating vegetation layers (Section 3.2), we first perform the 2D simulation of harmonic motion at the simulation plane (Section 3.3). We also propose a procedural method to project the simulation results if the simulation plane is not parallel to the viewing plane in order to retain the viewing coherence (Section 3.4). This 2D simulation results is used for modeling the vegetation's dynamics at the first layer, which is then utilized to guide the dynamics of other parts of the vegetation (Section 3.5). Based on the dynamics of vegetation, we animate the 3D vegetation model layer by layer (Section 3.6).

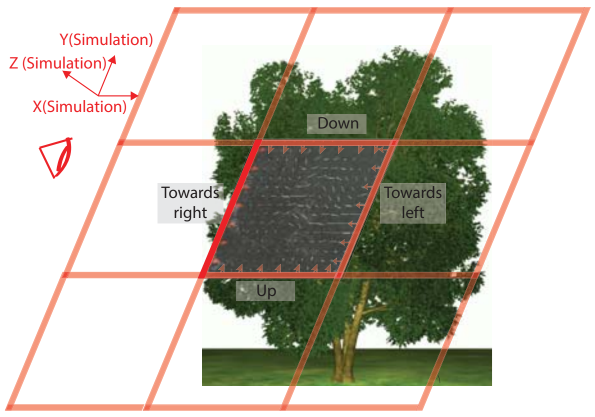


Figure 4: *The wind field and nine tiles of the simulation plane.*

3.2 Creating vegetation layers

We segment the vegetation's 3D "polygon soup" into clusters parallel to the viewing plane, and each cluster of meshes is rendered into one vegetation layer, which forms one animation component of our system (Figure 3). Vegetation layers are created dynamically as the viewer moves. The 3D model of vegetation usually contains the material or texture information for branches and leaves. We segment out the rigid branches and trunks (based on their materials or textures, for example the branch and leaf should have different texture) and consider them as the background. If the branches are soft, we can treat them as leaves. The segmentation can be obtained via rendering by setting near and far planes based on the vertices' depth (namely simple segmentation method).

The simple segmentation method is efficient. However it may cause some leaves be cut and located at two different layers. We observed that for vegetation with very small leaves this seam is not obvious, hence the simple segmentation is sufficient. However for vegetation with larger leaves, it causes noticeable artifacts. Thus, we propose to improve the segmentation method by pre-processing the 3D vegetation model. For the input "polygon soup" excluding rigid branches and trunks, we perform a breadth first search (BFS) at each vertex to find and group all vertices that are connected with it. The BFS is based on the neighboring information derived from the input mesh. This group of vertices is considered as one leaf which usually consists of one or several connected polygons. Since the leaves and branches are segmented apart, the leaves are disconnected from each other, therefore performing a BFS can isolate one leaf. If the texture coordinates (uv) are available, we choose the vertex with the smallest u and v coordinates as the representative (or root) of the leaf, otherwise we can choose one arbitrary vertex. We proceed this BFS to the next unvisited vertex until all the vertices are visited. After this pre-processing step, the root vertex of each leaf is obtained. At run time, when we perform the rendering by setting near and far planes, for each vertex, we pass one additional index array to the vertex shader which is the index to the root of the leaf that the vertex belongs to. The near and far planes are compared against only the root vertices, consequently no leaf is cut. Note that the pre-processing needs to be done only once.

The number of layers usually is set based on the type of vegetation scene and its distance to the viewer. For a large scale vegetation scene or when the vegetation scene is near to the viewer, more vegetation layers are required. In our implementation, we also apply a simple LOD method by dividing the vegetation scene into near, mid, and far regions relative to the distances to the viewer. Users can specify the distance of these regions and the number of vegetation layers used to represent each region. Inside one region, the distances between vegetation layers is the same. In general, there is one optimum number of vegetation layers which can achieve a natural looking animation for a vegetation scene. Beyond this number, only small improvement in animation quality can be observed. In practice, we always seek this optimum number of layers to achieve the trade off between quality and cost (Section 4). After each vegetation layer is animated, we compose them with the background (including the rigid branches and trunks) using alpha matting with the help of the depth map of the vegetation scene.

3.3 2D simulation of harmonic motion

The simulation plane to perform 2D harmonic motion simulation is defined by a user using a viewing plane by specifying one camera setting with location, orientation and FOV properties. For example, we can use the initial viewing plane as the simulation plane. We observed that wind effects occurred at different parts have some similarities, thus we perform a 2D simulation only on the 2D screen space which contains the visible region of the simulation plane when the user defines it. We then tile the rest of the simulation plane using this 2D simulation result (Figure 4). The other tiles are needed in case the view changes to show more by sampling method. We adopt a directional homogenous wind field, that is we assume the wind velocity is the same in planes parallel to the simulation plane. In other words, it can conceptually be considered as stacking the tiled 2D simulation result in the direction parallel to the simulation plane to approximate the wind effects in 3D space.

This 2D simulation of harmonic motion models the dynamics of vegetation at the first layer. If the viewer's viewing plane is not parallel to the simulation plane, a projection for the 2D simulation plane needs to be performed to determine the dynamics of vegetation at the first layer (Section 3.4). This dynamics is then used to approximate the dynamics of other vegetation layers (Section 3.5). The 2D simulation of harmonic motion (based on [Chen and Johan 2013]) includes a 2D wind simulation and a harmonic oscillator. The 2D wind simulation computes the wind's velocity $\vec{V}_{s_{wind}}$ (two dimensions and in simulation coordinates). $\vec{V}_{s_{wind}}$ is used to initiate the harmonic oscillator (continuum simulation) which computes the vegetation's velocity $\vec{V}_s$ (two dimensions and in simulation coordinates) at the simulation plane.

2D wind simulation: We use the 2D wind simulation approach in [Chen and Johan 2013] which applies the solver from [Stam 1999] to solve the Navier-Stokes equations in order to compute wind's velocity $\vec{V}_{s_{wind}}$.

2D harmonic oscillator: In general, the fluid simulation result lacks the oscillation effects. The oscillator is required to model these effects to mimic the swaying motion of vegetation. We consider a vegetation layer as one continuum represented using a 2D grid. We adopt the wave simulation to model the dynamics of continuum as in [Chen and Johan 2013]. Each grid cell, which is one simulation element, represents one or some vegetation parts (such as leaves). We consider the simulation element's velocity ($\vec{V}_s$, at the simulation plane) as the wave amplitude which is initiated by the wind's velocity ($\vec{V}_{s_{wind}}$) and apply the wave equation and then Verlet integration to compute and update the simulation element's velocity $\vec{V}_s$ as in [Chen and Johan 2013].

Initiating 2D simulation: This 2D simulation and user's initialization are all at the simulation plane. Users can set wind sources at the boundaries of the 2D wind simulation: leftward, rightward, up or down, to determine the wind direction. For example, if the viewer is viewing the vegetation from top, setting leftward, rightward, up and down motions in the simulation plane initiates the vegetation's leftward, rightward, frontward and backward motions accordingly. Users can also set the frequency and strength to automatically inject wind into the simulation.

Tiling 2D simulation: One possible solution is to use the periodic

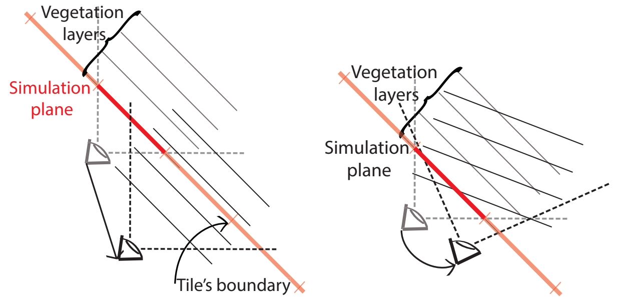


Figure 5: *Simulating dynamics of first vegetation layer, In (a) and (b), the simulation plane is set using the initial viewing plane. (a) The dynamics of first layer is the same in planes parallel to the simulation plane. (b) The dynamics of first layer needs to be projected.*

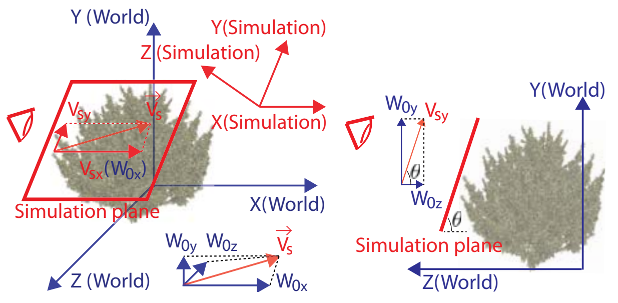


Figure 6: *Projection of the dynamics of vegetation from the simulation coordinates to world's coordinates. Only one tile of simulation plane is shown. The red viewer defines the simulation plane. (a) Perspective view. (b) Side view.*

boundary conditions to create tilable simulation. In our paper, we adopt a method, which is commonly used to make tilable textures as in [Chen et al. 2013], to make the 2D simulation result tilable by offsetting the 2D simulation result, wrapping around by half, and linearly blending the seams. The simulation results using other 2D simulation methods such as [Ramraj Kim Pallister, 2005] can be applied and made tilable to model the dynamics of vegetation.

3.4 Simulating dynamics of first vegetation layer

When the vegetation layers are parallel to the simulation plane, the X and Y components ($\vec{V_0}_{xy}$) of the vegetation's velocity at the first layer ($\vec{V_0}$) is obtained from the 2D simulation result: $\vec{V_0}_{xy} = \vec{V_s}$ (Figure 5 (a)). The Z component (V_{0_z}) of $\vec{V_0}$ is initialized as 0 and it is not explicitly determined by the 2D simulation. Its value will be updated in the following projection step and its effects (motion effects in depth) also include the propagation effects among vegetation layers in Section 3.5 and vertically shrinking the vegetation elements in Section 3.6.
When the vegetation layers are not parallel to the simulation plane, a projection is required to compute the dynamics of the first layer (Figure 5 (b)). (1) As in Figure 6, based on the simulation plane's position and orientation in world's coordinates, we project the $\vec{V}_{s_{xy}}$, from the simulation coordinates to the world's coordinates $\vec{W}_{0_{xyz}}$:

$$W_{0_x} = V_{s_x},\quad W_{0_y} = V_{s_y}\sin\theta,\quad W_{0_z} = V_{s_y}\cos\theta. \tag{1}$$

(2) As in Figure 7(a), if the vegetation layers and the simulation plane has an angle α in XZ plane, $\vec{W}_0$'s Y component W_{0_y} does not

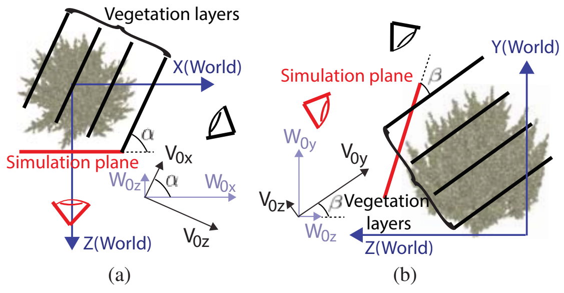


Figure 7: *Projection of the dynamics of vegetation from the world's coordinates to viewing coordinates. Only one tile of simulation plane is shown. The red viewer defines the simulation plane. The vegetation layers and the simulation plane (a) has an angle α in XZ plane (top view) (b) has an angle β in YZ plane (side view).*

change and we decompose $\vec{W}_0$'s X and Z components in viewing coordinates:

$$\begin{aligned} V_{0_x} &= W_{0_x}\cos\alpha + W_{0_z}\sin\alpha,\quad V_{0_y} = W_{0_y}, \\ V_{0_z} &= W_{0_x}\sin\alpha + W_{0_z}\cos\alpha. \end{aligned} \tag{2}$$

$\vec{V_0}$ is the vegetation's velocity at the first vegetation layer in viewing coordinates. We handle the cases if the vegetation layers and the simulation plane has an angle β in YZ plane (Figure 7(b)) or an angle in XY plane in a similar way.

3.5 Guided dynamics of other vegetation layers

To compute the dynamics for the remaining vegetation layers, one approach is to perform the 2D simulation of harmonic motion and compute the vegetation's dynamics layer by layer. However, for real-time applications such as games, less computational cost is important. Moreover, wind effects at different layers have some similarities. Thus, to reduce computational cost we propose to use the vegetation's dynamics at the first layer to approximate the animation of the remaining vegetation layers. Furthermore, the motion effects in depth are one important visual cue for the viewer to feel 3D motion effects. Thus, we propose to simulate the layer-layer interaction which captures the vegetation movement's propagation in the direction perpendicular to the viewing plane.
Firstly, we consider the case when the wind is blowing from front to back ($V_{0_z} < 0$ perpendicular to the viewing plane). Based on the observation, the vegetation at the front moves first, then, the vegetation's movements propagate backwards from the front to the back. In other words, the back vegetation layers react later than the first vegetation layer and the movements ($\vec{V}$) of the back layers can be considered as the previous movements ($\vec{V}$) of the first layer. Basically, we back track in time at the first layer to compute the vegetation's velocity at the other layers. $\vec{V_i}$ represents vegetation's velocity at i-th layer starting from 0.
For a vegetation representation with n layers 0 to $n-1$ at position A, where $\{L_0 \ldots L_{n-1}\}$ are the distances to the layer 0 for each vegetation layer, the velocity of vegetation at time t is $\left\{\vec{V_0}(t)_A \ldots \vec{V}_{n-1}(t)_A\right\}$:

$$\begin{aligned} \vec{V_i}(t)_A &= \vec{V_0}(t-\gamma_i)_A,\ (0 \le i \le n-1), \\ \gamma_i &= L_i/(-V_{0_z}(t)_A + V_D), \end{aligned} \tag{3, 4}$$

where V_D is the factor to control the time difference (γ_i) when the back layers start to move relative to the time when the vegetation in the first layer moves. γ_i is computed based on the propagation

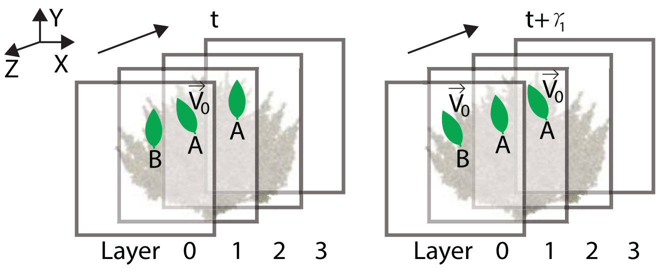


Figure 8: *During time γ_1, the movement propagates from position A to B at layer 0. The movement also propagates from layer 0 to 1 at the position A.*

time in viewing direction. If $t-\gamma_i$ does not align with the timestep boundary, we apply a linear interpolation to compute the velocity. V_D is set to create an offset of motions exhibiting at each layer. These motion's differences between layers generate occlusion effects in the vegetation animation. In practice, it can be set as a small random number, or in the case that the users want to initiate vegetation's velocity in the direction perpendicular to the simulation plane (in Section 3.3, wind only initiates the vegetation' velocity parallel to the simulation plane), it can be set as a constant number. If there is no restriction on memory consumption, we can store the dynamics of vegetation at the first layer at several time steps and use them to evaluate Equation (3).

Memory cost reduction: In real-time applications such as games, less memory consumption is important. Thus, we also propose a method to reduce the memory cost. Instead of storing many previous states of vegetation, we store only the latest result of the dynamics of vegetation at the first layer and perform an approximate computation as follows.

The 2D simulation method basically propagates the vegetation's movements in the 2D simulation plane while oscillating them. In a dense vegetation field, the vegetation usually distributes uniformly in all orientations. We assume one vegetation part affects nearby part in the Z direction in the same way as it propagates its movements in the XY plane. In Figure 8, consider the movement ($\vec{V_0}$) propagates from position A to B at layer 0 during time γ_1. The velocity of vegetation ($\vec{V_0}$) at the position B of layer 0 is actually the velocity of vegetation ($\vec{V_0}$) at the position A of layer 0 at γ_1 time ago. The direction of the movement's propagation at the position A of layer 0 is $\frac{\vec{V}_{0_{xy}}(t)_A}{\|\vec{V}_{0_{xy}}(t)_A\|}$, where $\vec{V}_{0_{xy}}(t)_A$ is the X and Y components of $\vec{V_0}(t)_A$. For the i-th vegetation layer, the vegetation velocity at the position A at time t, $\vec{V_i}(t)_A$ is approximated as

$$\vec{V_i}(t)_A = \vec{V_0}(t-\gamma_i)_A \approx \vec{V_0}(t)_B, B = A + \gamma_i s \vec{V}_{0_{xy}}(t)_A, \quad (5)$$

where s is the wave propagation speed. $\gamma_i s \vec{V}_{0_{xy}}(t)$ approximates the distance the movement travels during γ_i. In practice, only one look up (sampling) operation is required to compute the dynamics of vegetation at one position for the back layers.

In the case when the wind is blowing from back to front ($V_{0_z} > 0$, along the direction perpendicular to the viewing plane), the back vegetation layers react ahead than the first vegetation layer and the movements of the back layer can be considered as the future movements of the first layer. Based on Equation (3), γ_i is less than 0, thus $\vec{V_i}(t)_A$ is approximated in the opposite direction of the movement's propagation at A of layer 0, $-\frac{\vec{V}_{0_{xy}}(t)_A}{\|\vec{V}_{0_{xy}}(t)_A\|}$.

3.6 Animating vegetation layers

To generate the motion of vegetation, we define a uniform 2D grid over a vegetation layer and apply inverse warping to deform it as

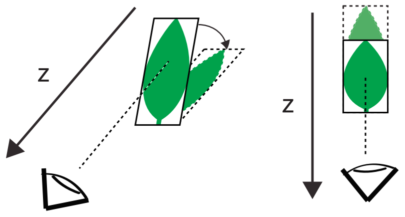


Figure 9: *The bending along Z direction appears as vertically shrinking.*

in [Chen and Johan 2013]. At each frame, for each vegetation layer i, we compute the amount of displacement $\vec{d_i}$ of the grid' corners using the vegetation's velocity $\vec{V_i}$ in the current viewing coordinates, which is obtained by bi-linear sampling from the simulation result in Section 3.3 and using the approximation method in Section 3.5. We extend the method in [Chen and Johan 2013] to model the movements occurring along the direction perpendicular to the viewing plane (either towards or away from the viewer). As in Figure 9, these movements appear as the vegetation element is shrinking vertically. This is also helpful to mimic the motion effects in depth. It is realized by stretching the underlying 2D grid cells, since the deformation is based on inverse warping. We displace the grid's corners along Y direction with $\lambda_z |V_{i_z}(t)|$, which models the shrinking effect generated by the V_{i_z}.

$$\begin{aligned} d_{i_x} &= R\left(1-\left(\frac{L_i}{L_{n-1}}\right)^2\right) m\lambda_x V_{i_x}(t)\Delta t, \quad (6) \\ d_{i_y} &= R\left(1-\left(\frac{L_i}{L_{n-1}}\right)^2\right) m\left(\lambda_y V_{i_y}(t) - \lambda_z |V_{i_z}(t)|\right)\Delta t, \end{aligned}$$

$$R = \begin{cases} 1 & \text{if every parts can move,} \\ \left(\frac{l}{h}\right)^r & \text{otherwise,} \end{cases} \quad (7)$$

where $\vec{\lambda}$ is a scale factor to control the amount of displacement $\vec{d_i}$, Δt is the simulation time step, $\vec{\lambda}$ is set with respect to the grid cell's size, l is the distance to the root position of vegetation layer, h is the height of the vegetation layer, and $r \in [0,10]$ is the rigidity parameter that controls the amount of displacement with respect to the distance to the vegetation's root position, whose position is fixed. We compute the bounding box of the vegetation and use the bottom of the bounding box as the root position. On screen space, the vegetation further from the viewer exhibits less movement than the vegetation closer to the viewer. Thus we introduce $\left(1-\left(\frac{L_i}{L_{n-1}}\right)^2\right) m$, $(0 \le i \le n-1)$ to reduce the deformation, n is the number of layers, $m \in [0,1]$ defines the maximum percentage to be deducted, m is greater for a larger vegetation scene.

4 Results and discussion

We applied our proposed method to animate dense 3D vegetation: shrub, tree crown, grass field, underwater vegetation and vegetation field (Figure 1, also please refer to the supplementary video). Our method was implemented on a desktop PC with Intel Xeon E5520 2.27Ghz CPU, 4.0G Memory and Nvidia GeForce GTX275 GPU. In our experiments, the 2D fluid simulator (with a 64 by 64 grid [Stam 1999]) and the harmonic oscillator were implemented in the CPU, while the rest of the steps were implemented in the GPU. We believe shifting the whole implementation into GPU and using

Table 1: *Performance measurement*

No.faces	No.layers	FPS
80K	3(1-1-1)	70
80K	5(3-1-1)	65
50M	3(1-1-1)	40
50M	5(3-1-1)	34

a better GPU than what we used in our experiments could further increase the frame rate. With a frame resolution of 1024 x 768, we measure the FPS for two vegetation meshes/scenes with different number of faces and with different number of vegetation layers as shown in Table 1. The numbers in brackets indicate the number of vegetation layers to represent near-mid-far regions respectively. We observed that even if the number of faces of vegetation increases a lot, the application is still able to achieve real-time frame rate. This is because, with the same number of vegetation layers, the simulation cost always remains constant, only the segmentation (rendering) cost leads to the drop in frame rate.
Our method demonstrates that for dense vegetation, using the proposed simple billboard layers representation and utilizing 2D animation methods can achieve natural looking animation results. By doing so, the simulation is only required in 2D screen space, thus it is very efficient. Furthermore, the visual cues (motion effects in depth and occlusion effects) play an important role to make up for the main shortcoming of 2D methods which is the lack of information to perceive the motion in 3D. Users can specify the distances of the near, mid and far regions and the number of vegetation layers used to represent each region. In our experiments, we usually use 3 (smaller vegetation) to 5 (larger vegetation scene as the last scene in Figure 1) vegetation layers. For the last scene in Figure 1, we found that using 5 layers can achieve the balance between the animation quality and computational cost. We also tried to increase the number of vegetation layers, such as to 7, however the improvement in the quality of animation is less noticeable.
Our method can achieve real-time frame rate, and as a result users can change the values of parameters on the fly to adjust the animation. Our method does not rely on the type of input, it can handle 3D models such as low resolution polygons or in volumetric format (vegetation layer can be generated from the slices) using the simple segmentation method, which is usually difficult to set up underlying animation structures using existing methods.
Comparisons: We evaluate our method by comparing our results with real vegetation and the wind effects of the state-of-the-art award-winning toolkit [SpeedTree] (please refer to the supplementary video). We observe in real vegetation's motion and in the [SpeedTree] Cinema (not for real-time applications), vegetation elements may group and move together and the motion effects in depth as well as occlusion effects also happen. Our approach can generate these effects and simulate the harmonic motion of vegetation in wind such as swaying and oscillating.
We also compare our results with a PC game [Sword Heroes' Fate 3] using [SpeedTree] for Games. With its sophisticated rendering LOD and wind LOD system, a large scale vegetation field can be animated in real-time. Users can specify the behaviors of vegetation and the properties of the procedural wind by setting the parameters and the underlying animation model can be automatically embedded. However, [SpeedTree] for Games tends to animate each single element in 3D, and in run-time per-vertex wind data or weighted skeleton information need to be computed (or fetched) per-instance. On the other hand, we demonstrate that for dense vegetation field, focusing on the overall appearances and the apparent parts of vegetation's motions is sufficient to generate natural looking animation, while the simulation is only required in 2D screen space without skeleton-based computation. As a result, the computational cost can be reduced and this is important since we can allocate more computational resources to other processing, such as AI.
Limitations: Some non-correlated vegetation elements may group and move together. We can increase the number of vegetation layers to reduce the undesired grouping effects. Our method may perform less well if the vegetation has obvious rigid branches. Our method is suitable for animating dense vegetation scene where the movements of tree's rigid trunk and branches are usually not obvious especially when the tree is swaying in a light breeze. The proposed billboard layer structure is simple, but it is not sufficient to capture the twisting or flipping effects of leaves under the influence of wind in which cases the back side of leave may appear. Thus, our method is not suitable for simulating vegetation with sparse and very big leaves such as palm. Our method performs better for vegetation with smaller leaves, or dense vegetation oscillating in a light breeze. In these cases less non-twisting artifacts can be noticed.

5 Conclusion and future work

In this paper, we have presented a 2D method to animate 3D vegetation in real-time. Our method is suitable to simulate dense vegetation under the influence of wind. We segment the vegetation meshes into view-dependent 2D billboard layers and perform a 2D simulation of harmonic motion at a simulation plane to model the dynamics of the first layer. We also propose a procedural method to project the simulation result when the vegetation layers do not parallel to the simulation plane. Then, we utilize the vegetation's dynamics at the first layer to guide the dynamics of the other layers. We also consider the motion effects in depth and occlusion effects. As a result, we can animate shrub, tree crown, grass swaying in wind as well as aquarium plant swaying in water flow in real-time. However our method does not require tedious work to set up the underlying animation structure or performing an explicit 3D simulation. One future work is to investigate other types of 2D animation methods besides the grid-based method to handle more types of vegetation and effects. In addition, we want to improve the current shading and incorporate ambient occlusion. We also plan to animate other natural elements in 3D using 2D billboard layers, such as smoke and cloud.
Acknowledgements We gratefully thank the reviewers for their constructive comments. This research was supported by Fraunhofer IDM@NTU, which is funded by the National Research Foundation and managed through the multi-agency Interactive & Digital Media Programme Office hosted by the Media Development Authority of Singapore. The input meshes are from [Xfrog], [SpeedTree] and [Archmodels].

References

AKAGI, Y., AND KITAJIMA, K. 2006. Computer animation of swaying trees based on physical simulation. *Computers and Graphics 30*, 4, 529–539.

ARCHMODELS. Evermotion, Inc.

BANISCH, S., AND WUTHRIC, C. A. 2006. Making grass and fur move. *Journal of WSCG 2006 14*, 25–32.

BAO, G., ZHANG, X., CHE, W., AND JAEGER, M. 2009. Billboards for tree simplification and real-time forest rendering. In *Plant Growth Modeling, Simulation, Visualization and Applications (PMA), 2009*, 433–440.

BEAUDOIN, J., AND KEYSER, J. 2004. Simulation levels of detail for plant motion. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

BEHRENDT, S., COLDITZ, C., FRANZKE, O., KOPF, J., AND DEUSSEN, O. 2005. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum 24*, 3, 507–516.

BRUNETON, E., AND NEYRET, F. 2012. Real-time realistic rendering and lighting of forests. *Computer Graphics Forum 31*, 2pt1, 373–382.

CHEN, K., AND JOHAN, H. 2010. Real-time continuum grass. In *VR '10: Proceedings of the 2010 IEEE Virtual Reality Conference*, 227–234.

CHEN, K., AND JOHAN, H. 2013. A simple method to animate vegetation in images using simulation-guided grid-based warping. In *CAD/Graphics '13: Proceedings of the 14th International Conference on Computer-Aided Design and Computer Graphics*, 244–251.

CHEN, K., JOHAN, H., AND MUELLER-WITTIG, W. 2013. Simple and efficient example-based texture synthesis using tiling and deformation. In *I3D '13: Proceedings of 2013 Symposium on Interactive 3D Graphics and Games*, 145–152.

CHUANG, Y.-Y., GOLDMAN, D. B., ZHENG, K. C., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2005. Animating pictures with stochastic motion textures. *ACM Transactions on Graphics 24*, 3, 853–860.

DECAUDIN, P., AND NEYRET, F. 2004. Rendering forest scenes in real-time. In *Rendering Techniques '04: Proceedings of Eurographics Symposium on Rendering*, 93–102.

DECAUDIN, P., AND NEYRET, F. 2009. Volumetric Billboards. *Computer Graphics Forum 28*, 8, 2079–2089.

DÉCORET, X., DURAND, F., SILLION, F. X., AND DORSEY, J. 2003. Billboard clouds for extreme model simplification. *ACM Transactions on Graphics 22*, 3, 689–696.

DIENER, J., RODRIGUEZ, M., BABOUD, L., AND REVERET, L. 2009. Wind projection basis for real-time animation of trees. *Computer Graphics Forum 28*, 2, 20–28.

GUERRAZ, S., PERBET, F., RAULO, D., FAURE, F., AND CANI, M.-P. 2003. A procedural approach to animate interactive natural sceneries. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents*, 73–78.

HABEL, R., WIMMER, M., AND JESCHKE, S. 2007. Instant animated grass. *Journal of WSCG 2007 15*, 123–128.

HABEL, R., KUSTERNIG, A., AND WIMMER, M. 2009. Physically guided animation of trees. *Computer Graphics Forum 28*, 2, 523–532.

HU, S., FUJIMOTO, T., AND CHIBA, N. 2009. Pseudo-dynamics model of a cantilever beam for animating flexible leaves and branches in wind field. *Computer Animation and Virtual Worlds 20*, 2-3, 279–287.

HU, S., CHIBA, N., AND HE, D. 2012. Realistic animation of interactive trees. *The Visual Computer 28*, 6-8, 859–868.

JENSN, O., SALAMA, C. R., AND KOLB, A. 2009. GPU-based responsive grass. In *Journal of WSCG 2009*, UNION Agency - Science Press.

KAJIYA, J. T., AND KAY, T. 1989. Rendering fur with three dimensional textures. *Computer Graphics (Proceedings of ACM SIGGRAPH '89) 23*, 3, 271–280.

LENGYEL, J., PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2001. Real-time fur over arbitrary surfaces. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, 227–232.

MEYER, A., AND NEYRET, F. 1998. Interactive volumetric textures. In *Rendering Techniques '98: Proceedings of Eurographics Symposium on Rendering*, 157–168.

OTA, S., FUJIMOTO, T., TAMURA, M., MURAOKA, K., FUJITA, K., AND CHIBA, N. 2003. 1/f β noise-based real-time animation of trees swaying in wind fields. In *CGI '03: Proceedings of Computer Graphics International, 2003.*, 52–59.

PELZER, K. Randima Femando, 2004. Rendering countless blades of waving grass. *GPU Gems*, 107–121.

PERBET, F., AND CANI, M.-P. 2001. Animating prairies in real-time. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, 103–110.

RAMRAJ, R. Kim Pallister, 2005. Dynamic grass simulation and other natural effects. *Game Programming Gems 5*, 411–419.

REEVES, W. T., AND BLAU, R. 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *SIGGRAPH '85: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, 313–322.

SHINYA, M., AOKI, M., TSUTSUGUCHI, K., AND KOTANI, N. 1999. Dynamic texture: physically based 2d animation. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 239.

SOUSA, T. Hubert Nguyen, 2007. Vegetation procedural animation and shading in Crysis. *GPU Gems 3*, 373–385.

SPEEDTREE. Interactive Data Visualization, Inc.

STAM, J. 1997. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum 16*, 3, 159–164.

STAM, J. 1999. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 121–128.

SWORD HEROES' FATE 3. Kingsoft Corporation.

WANG, C., WANG, Z., ZHOU, Q., SONG, C., GUAN, Y., AND PENG, Q. 2005. Dynamic modeling and rendering of grass wagging in wind: Natural phenomena and special effects. *Computer Animation and Virtual Worlds 16*, 3-4, 377–389.

WEBER, J. 2008. Fast simulation of realistic trees. *Computer Graphics and Applications, IEEE 28*, 3, 67–75.

XFROG. Xfrog, Inc.

ZHANG, L., SONG, C., TAN, Q., CHEN, W., AND PENG, Q. 2006. Quasi-physical simulation of large-scale dynamic forest scenes. In *CGI'06: Proceedings of the 24th international conference on Advances in Computer Graphics*, 735–742.

ZHANG, L., ZHANG, Y., JIANG, Z., LI, L., CHEN, W., AND PENG, Q. 2007. Precomputing data-driven tree animation. *Computer Animation and Virtual Worlds 18*, 4-5, 371–382.

ZHAO, Y., AND BARBIČ, J. 2013. Interactive authoring of simulation-ready plants. *ACM Transactions on Graphics 32*, 4, 84:1–84:12.

ZIOMA, R. Hubert Nguyen, 2007. GPU-generated procedural wind animation for trees. *GPU Gems 3*, 105–120.