Real-time animation of vegetation : novel 2D approach and interactive authoring interface

Chen, Kan

2017

Chen, K. (2017). Real-time animation of vegetation : novel 2D approach and interactive authoring interface. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/69490>

<https://doi.org/10.32657/10356/69490>

NANYANG
TECHNOLOGICAL
UNIVERSITY

Real-time Animation of Vegetation: Novel 2D Approach and Interactive Authoring Interface

by

Chen Kan

School of Computer Science and Engineering

Nanyang Technological University

2017

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

Real-time Animation of Vegetation: Novel 2D Approach and Interactive Authoring Interface

by

Chen Kan

A thesis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

in the

School of Computer Science and Engineering

Nanyang Technological University

2017

Abstract

Real-time animation of vegetation (a field of plants), e.g., grass, shrubs, and trees, has many applications such as in movies, cartoons, and games. However, animating a vegetation scene is a challenging problem, because in general a vegetation scene has complex geometry and may consist of large quantity of elements (grass blades, leaves etc.). The conventional method of manual drawing (e.g., 2D cartoons) or setting poses of plants at many key frames (e.g., 3D animation movies) requires tedious work and art skills. Physics-based methods can achieve physically realistic animation, but they require high computational costs and lack intuitive direct control. This thesis presents novel algorithms to efficiently animate vegetation in 2D images and 3D scenes in real-time as well as provide users with an intuitive interface to easily and rapidly design and generate animation of short plants. This work focuses on modeling the motions of plants under the influence of external forces, e.g., natural wind and passing object.

Based on observations, vegetation elements may stick together and move in groups, exhibiting harmonic motion effects such as deforming and oscillating. To realize these effects, a real-time method has been proposed, which combines fluid simulation, wave simulation, and grid-based image warping to simulate harmonic motions of vegetation in 2D images, in particular cartoon style images.

For real-time 3D applications such as games, it is important to lower the computational cost of vegetation animation while still making the animation looks visually plausible. Therefore, an efficient 2D-based approach for real-time animation of vegetation in 3D scenes has been proposed, especially suitable for vegetation with densely leaved foliage. This approach represents vegetation as view-dependent 2D billboard layers, adopts the proposed 2D harmonic motion simulation for modeling the dynamics of vegetation at the

first layer (closest to the viewer), and utilizes this dynamics to guide the animation of the rest of the layers while addressing the motion effects in depth and occlusion effects.

Lastly, short plants, such as grass, flowers, and shrubs, are common vegetation species. It is not always easy or intuitive for users to generate their desired motions of short plants using existing methods. Thus, a novel approach to author the common motions of short plants which are bending and twisting has been proposed. It is based on the observation that hand motions can represent the bending and twisting motions of short plants and using hand to describe motions is natural and proficient for human. Therefore, a hand is used as a ‘puppet’ to author the animation of one single short plant. First, the global motions of the short plant are authored, followed by the motions of its elements such as leaves. A framework to utilize the animation results to animate a field of short plants has also been proposed.

Using the proposed methods, the manual efforts of animators and the computational costs can be significantly reduced, and natural-looking vegetation animation (comparable with real videos, cartoons, and the results using existing commercial software) can be achieved. The proposed 2D method can be applied to enhance vegetation photographs or paintings with animated vegetation motions as well as to help animators to create cartoon animations of vegetation. Additionally, this method can be incorporated with haptic interaction to provide users with a multisensory experience with images. The proposed 3D animation method allows dense vegetation scenes to be animated in real-time applications. The proposed animation authoring method empowers users to intuitively and rapidly author and generate their desired motions of short plants.

Keywords: 2D/3D vegetation, real-time animation, harmonic motion, 2D billboards, animation authoring, hand gestures

Acknowledgements

I would like to thank my supervisor, Dr. Henry Johan, for his aspiring guidance and encouragement throughout my study as a postgraduate student in Nanyang Technological University.

I also would like to thank Assoc Prof. Alexei Sourin, for his help on managing the project. Many thanks go to Fraunhofer IDM@NTU and Assoc Prof. Wolfgang Müller-Wittig, for providing an excellent work environment for this research.

Furthermore, I would like to thank all my friends and colleagues in Fraunhofer IDM@NTU for their valuable help and fruitful discussions on this research, in particular, Dr.-Ing. Marius Erdt, Mr. Ajin Joseph, Mr. Koa Ming Di, Mr. Eugene Lee, Mr. Junming Peng, Dr. Qi Ming, Dr. Shahzad Rasool, Mr. Adrian Soh, Dr. Sun Qian, Dr. Budianto Tandianus, Mr. Gerrit Voss, Mr. Nicholas Mario Wardhana, and Ms. Zhu Chao.

Finally and above all I would like to express my deepest gratitude to my family. I am grateful to my mother-in-law Huang Wei, father-in-law Han Wei, my mother Wang Jian and my father Chen Feiqiang for their love and extraordinary support. Moreover, I am very thankful to my wife Han Jing for her love and sacrifices over the years. I am also thankful to my daughter Chen Xin for being such a good girl always cheering me up.

Contents

Abstract	ii
Acknowledgements	iv
Table of Contents	v
List of Figures	viii
List of Tables	xiii
1 Introduction	1
1.1 Background	1
1.2 Objectives	9
1.3 Contributions	11
1.4 Report organization	15
2 Related Work	16
2.1 Simulation of nature scene	16
2.1.1 Fluid	17
2.1.2 Terrain	18
2.1.3 Wind	19
2.2 Animation of vegetation in 2D images	20
2.2.1 Animating 2D still images	20
2.2.2 Simulation in cartoons	22
2.2.3 Summary of animation of vegetation in 2D images	23
2.3 Animation of vegetation in 3D scenes	25
2.3.1 Modeling of grass	25
2.3.2 Animation of grass	28
2.3.3 Modeling of trees	31
2.3.4 Animation of trees	36
2.3.5 Hair and fur	39
2.3.6 Summary of animation of vegetation in 3D scenes	40

3	Animating Vegetation in 2D Images Using Simulation-guided Grid-based Warping	43
3.1	Basic idea	44
3.2	Segmentation and compositing	46
3.3	Wind model	48
3.4	Simulating dynamics of vegetation	49
3.5	2D vegetation animation	50
3.6	Results and discussion	54
3.6.1	Comparisons and analysis of results	57
3.6.2	Limitations	59
3.6.3	Generalization	61
3.7	Summary	64
4	Animating 3D Vegetation in Real-time Using a 2D Approach	70
4.1	Basic idea and steps	71
4.2	Creating vegetation layers	73
4.3	2D simulation of harmonic motion	76
4.4	Simulating dynamics of first vegetation layer	80
4.5	Guided dynamics of other vegetation layers	82
4.5.1	Memory cost reduction	83
4.6	Animating vegetation layers	85
4.7	Results and discussion	86
4.7.1	Comparisons and analysis of results	90
4.7.2	Limitations	93
4.8	Summary	94
5	Interactive Authoring of Bending and Twisting Motions of Short Plants Using Hand Gestures	100
5.1	Overview of the proposed system	101
5.2	Background of authoring motions of articulated object and hand gestures	107
5.2.1	Authoring and controlling motions of articulated objects	107
5.2.2	Hand gestures	108
5.3	Animation model	109
5.4	Animation authoring of a short plant using hand gestures	110
5.4.1	Bending	111
5.4.2	Twisting	113
5.4.3	Secondary animation authoring	114
5.4.4	Hand gesture capturing setup	115
5.5	Using and controlling the authored animations	116
5.5.1	Animation looping	116
5.5.2	Motion propagation	117
5.5.3	Animation control	119
5.5.4	Guided procedural animation	122
5.5.5	Alignment of secondary animation	123
5.6	Results and discussions	125

5.6.1	Comparisons and analysis of results	125
5.7	User study and evaluation	127
5.7.1	Generalization and limitations	131
5.8	Summary	132
6	Conclusion and Future Work	135
6.1	Conclusion	135
6.2	Future work	140
6.2.1	Animating vegetation in 2D images	140
6.2.2	Animating 3D vegetation	141
6.2.3	Applications and generalizations of the proposed methods	142
	Publications	145
	A Bi-cubic Interpolation Coefficients	146
	B Procedural Skinning Weights	148
	C User Study Questionnaire	149
	Bibliography	153

List of Figures

1.1	Screen shots of games (a) Crysis 3 (2013) [EA13] and (b) Uncharted 4: A Thief’s End (2016) [Dog16].	2
1.2	Vegetation is an essential component of a nature scene. Images are from www.flickr.com.	2
1.3	3D Production Pipeline [Bea12]. Pre-production: Idea, Story, Storyboard, Animatic, Design; Production: Layout, R&D, Modeling, Texturing, Rigging/Setup, Animation, VFX, Lighting, Rendering; Post-production: Compositing, 2D VFX/Motion Graphics, Color Correction, Final Output.	4
1.4	Movie frames from animation “Twins on the Pasture” (©CCTV Animation Inc.). Wind is blowing, but the vegetation in foreground remains still.	5
1.5	Our method is suitable for scenarios which require designing and simulating specific motions of short plants, such as in cartoons (a), (b, top) and modeling the motions of short plants under the influence of a passing character (b, top) or car (b, bottom, from www.youtube.com). Cartoon frame (a) is from animation “Kung Fu Panda 2” (©DreamWorks Animation), (b, top) is from “Naruto” (©Masashi Kishimoto/Hayato Date/Shueisha/Studio Pierrot/TV Tokyo/Viz Media).	7
2.1	Fluid simulation for wind [AK06].	18
2.2	Procedural method and wave simulator for wind [CJ10].	19
2.3	Segment a still image into layers and synthesize a motion texture to animate the image layer by layer. [CGZ ⁺ 05].	21
2.4	Synthesis video texture using patch-based texture synthesis based on graphcut. [KSE ⁺ 03].	21
2.5	Synthesis motion texture incorporating fluid simulation to animate under water plant. [MWGZ09].	22
2.6	Animate cartoon style tree based on building realistic 3D geometries of tree as underlying models. [DFVHVR03].	22
2.7	Particle system [RB85].	25
2.8	We [CJ10] proposed to use segmented billboards for grass modeling.	26
2.9	Shell grass [BH02].	27
2.10	A grass field with three levels of details [BPB06].	28
2.11	Grass-wind interaction based on free form deformation and explicit grass-grass collision detection [WWZ ⁺ 05].	29
2.12	Continuum grass. We consider grass field as a two dimensional grid-based continuum and shift the complex interactions to the dynamics of continuum [CJ10].	30

2.13	Plastic trees: interactive self-adapting botanical tree modeling [PSK ⁺ 12].	32
2.14	Image-based tree modeling [TZW ⁺ 07].	33
2.15	Billboard clouds tree model. [DDSD03].	34
2.16	Volumetric billboards [DN09].	35
2.17	LOD for realistic forest rendering [BN12].	36
2.18	Physics guided animation of trees [HKW09].	37
2.19	Wind projection basis for animation of tree [DRBR09].	37
2.20	Procedural animation of trees in game Crysis [EA13].	38
2.21	Animation of interactive trees based on harmonic oscillator and noises [HCH12].	38
3.1	The steps.	47
3.2	Wind field. The wind source is at the left boundary, and the wind direction is from left to right.	48
3.3	Grid-based warping. Left: Original vegetation layer with warped 2D grids. Right: Warped vegetation layer with 2D grids. Vegetation image is from www.colourbox.com (©Colourbox Educational).	51
3.4	Example of scale map. (a) Original vegetation scene. (b) Scale map, darker region means smaller magnitude for scale factor $\vec{\lambda}$. Vegetation image is from www.iwallscreen.com	52
3.5	Result animation using the proposed method using a cartoon (©Teemu Korhonen) as an input, the wind direction is from left to right. The most effected regions are highlighted in the second row.	54
3.6	Result animation using the proposed method using a photograph (©Moonglowlily) as an input, the wind direction is from right to left. (a) is one frame of the animated photograph sequence, for clarity, one part of the animated photograph sequence is enlarged and shown in (b), in (b) the most affected regions are highlighted in the second row.	55
3.7	Result animation using the proposed method using a sketch (©HUISJ.COM) as an input, the wind direction is from right to left. (a) is one frame of the animated sketch sequence, for clarity, one part of the animated sketch sequence is enlarged and shown in (b), in (b) the most affected regions are highlighted in the second row.	56
3.8	Result animation using the proposed method using a cartoon image from “The Adventures of Little Carp” (©CCTV Animation Inc.) as an input. The most effected regions are highlighted in the second row.	57
3.9	Parts of images exhibiting harmonic oscillations due to wind and fluid flow are highlighted: (a) trees and bushes, (b) water surface reflections, and (c) soft tissues inside the surgical scene being washed by the surgical fluid. The QR codes link to the actual videos published on YouTube.	61
3.10	Grid-based bi-cubic inverse image warping. Vegetation image is from www.colourbox.com (©Colourbox Educational).	62

3.11	Comparing bilinear interpolation with bi-cubic interpolation. (a) Bilinear interpolation with small deformation amount. (b) Bi-cubic interpolation with small deformation amount. (c) Bilinear interpolation with large deformation amount. (d) Bi-cubic interpolation with large deformation amount. In the last row, the most affected regions of our results are highlighted.	63
3.12	Comparing our result with a real cartoon “Natsume Yuujin-Chou” (“The Natsume’s Book of Friends”) (©Yuki Midorikawa/Nihon Ad Systems Inc./Natsume Yuujin-Chou Project). The wind direction is from bottom right to up left. In the last column, the most affected regions of our result are highlighted.	66
3.13	Comparing our result with a real cartoon “Twins on the Pasture” (©CCTV Animation Inc.). The wind direction is from bottom right to up left. In the last column of each example, the most affected regions of our result are highlighted.	67
3.14	Comparing our result with a real video (from www.youtube.com). The wind direction is from right to left. In the last column, the wave effect for the grass field of our result is highlighted.	68
3.15	Comparing our result with a real video (from www.youtube.com). The wind direction is from bottom right to up left. In the last column of each example, the most affected regions of our result are highlighted.	69
4.1	Our method can be used to animate (from left to right) shrub, tree, underwater vegetation, and vegetation scene.	71
4.2	The steps. Red: input. Green: main loop. Blue: update. Dotted boxes: the corresponding sections in this chapter.	72
4.3	Representation of a 3D shrub model. (a) The input 3D model. (b) Segment the model 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).	74
4.4	The wind field and nine tiles of the simulation plane.	77
4.5	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, and note that the simulation plane is static.	79
4.6	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.	80
4.7	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).	81
4.8	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	84
4.9	The bending along Z direction appears as vertically shrinking.	85

4.10	Result shrub animation using the proposed method, the wind direction is from left to right. Our method can simulate the deforming and oscillating effects as well as the occlusion effects. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.	87
4.11	Result shrub animation using the proposed method. Our method can handle camera rotation and model the motion effects in depth. The wind direction was originally from left to right and it is now from left back to right front respecting to the current camera location and viewing direction. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.	87
4.12	Result tree animation using the proposed method, the wind direction is from left to right. Our method can simulate the deforming and oscillating effects as well as the occlusion effects. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.	88
4.13	Result underwater plant animation using the proposed method, the water flow direction is from left to right. Our method can simulate the deforming effects as well as the occlusion effects. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.	89
4.14	Comparing our result with a real tree. In the last column, some deforming and occlusion effects are highlighted.	96
4.15	Comparing our result with a real shrub. In the last column, some deforming and occlusion effects are highlighted.	97
4.16	Comparing our result with a shrub animation created using SpeedTree Cinema (for movie applications). In the last column, some oscillation (vibration) and occlusion effects are highlighted.	98
4.17	Comparing our result with a tree animation created using SpeedTree Game (in a PC game Sword Heroes' Fate 3 [Swo]). In the last column, some deformation (swaying) and occlusion effects are highlighted.	99
5.1	Our method can be used to animate (first row, from left to right) grass, flower, shrub (two examples), (second row, from left to right) a scene of short plants (example of an 'CG' scene), and a scene of grass and short trees.	101
5.2	Transferring the motions of a hand to a short plant. The pitch and yaw motions of the wrist are transferred to the pitch and yaw motions of the root (joint 0) of the grass. They also determine the direction of bending for the upper parts (joints 1, 2, 3) of the grass. The roll motion of the wrist is transferred to the twisting of the upper parts. Bending angles of the finger joints control the magnitudes of the bending for the upper parts of the grass.	102

5.3	A hand can bend and roll which can represent the essential bending and twisting motions of short plants. Thus, it is natural to use a hand to author the motions of short plants. The followings are authoring examples of a grass blade. (a) Bending a hand forward and backward: grass bending forward and backward (pitch bending). (b) Bending a hand left and right: grass bending left and right (yaw bending). (c) Rolling a hand: twisting of grass.	103
5.4	Our system setup.	105
5.5	The steps. Dotted boxes: the corresponding sections in the chapter. We first do the rigging for the input 3D short plant model manually (Section 5.3). Then, we perform the animation authoring for one single short plant using hand gestures (Section 5.4). This authoring process is performed for the global animation first. This process may be repeated for the secondary elements (leaves). After authoring, the results of one short plant are utilized to animate the whole field (Section 5.5).	106
5.6	The underlying animation models for a grass blade and a flower. Their joints are labeled with respect to the joints of a hand. The yellow labels show the joints of the grass blade and the whole flower. The green labels show the joints of a secondary element (leaf) which is considered as a simple short plant.	109
5.7	Bending the upper parts of a short plant.	112
5.8	Authoring (a) the global animation for the whole flower, then (b) the secondary animation for the leaf.	114
5.9	Creating an animation sequence with infinite duration.	116
5.10	During time γ_B , the movement propagates from position A to B	118
5.11	User feedback. (a) Feedback of the system. (b) Feedback of the animation results. Please refer to Appendix C for the questionnaire and detailed feedback.	128
5.12	Motion similarity. (a) Cartoon wheat from “The Old Mill” from Walt Disney’s “Silly Symphonies”. (b) Our result. (c) Motion similarity chart.	130

List of Tables

4.1 Performance measurement	88
---------------------------------------	----

Dedicated to my parents.
谨以此文献给我的父亲母亲。

Chapter 1

Introduction

Cities cannot just be made up of concrete buildings, tarmac, and pavements. It would be depressing and unpleasant to live in. You need to balance that with trees and flowers. This will make Singapore more pleasant to live in.

Lee Kuan Yew

1.1 Background

In computer graphics, simulating nature scenes is one of the major research topics. It has various applications, such as digital games, movies, architectural visualization, and etc. Real-time simulation of a believable and interactive nature scene provides users with an immersive feeling (Figure [1.1](#)). Simulating nature scenes covers a wide range of research topics such as the simulation of fluid, terrain, wind, and vegetation. Vegetation is an

FIGURE 1.1: Screen shots of games (a) Crysis 3 (2013) [EA13] and (b) Uncharted 4: A Thief's End (2016) [Dog16].

FIGURE 1.2: Vegetation is an essential component of a nature scene. Images are from www.flickr.com.

essential component of a nature scene (Figure 1.2). The earth is covered by abundant types of vegetation such as grass, bush, shrub, flower, and tree. They can be found in valleys, forests, soccer fields, croplands, gardens, and etc.

With texturing and modeling techniques, detailed visual complexity of a vegetation scene can be realized (Figure 1.1). However, in order to present users with a rich view into a virtual environment, only visualizing the static scenery of vegetation is not sufficient. Never being unintentionally surprised by the motion or lack of it in a scene is one of the animation principles [TJ84]. Nowadays, an interactive and animated environment is becoming an essential part of digital games and animation: in the case of wind blowing

or object passing, water surface may be perturbed, a track may be left on the snow terrain, grass and trees should sway in wind and responsive to interactions. The users will feel more immersive if such natural behavior can be modeled. Not only being visually believable, interactive and animated vegetation scenes may also deliver essential insights of the scene such that the users can have a better understanding about the scene and the scenario. For example, when a character or a car moves across a grass field, the severely swaying grass makes it clear that the grass is soft rather than hard shrub, in addition, the character or the car should be moving in a relatively high speed rather than moving slowly. As such, vegetation animation is important, it makes the nature scene more believable and the users will be more satisfied.

In general, computer animation can be categorized into 2D and 3D animations. For 2D animation such as cartoons, the traditional approach is manually drawing the frames. In a conventional 3D animation pipeline (Figure 1.3), during the production phase the main steps are modeling, texturing, rigging, animation, and rendering [Bea12]. The modeling stage generates the 3D geometry of the model. The texturing stage applies texture on the model. The rigging stage establishes a control structure (handles) for the 3D models, then the animation stage generates poses of the rigged models by computing the shape based on the transformations of the handles. Finally, the final image is rendered on the screen at the rendering stage. The work in this thesis aims at proposing algorithms to handle both 2D and 3D vegetation animations.

It is difficult to model a vegetation scene, because it usually contains a huge number of leaves, branches, blades of grass, and foliage. Nowadays, vegetation rendering and modeling techniques are more and more popular. Due to the complexity of a vegetation

FIGURE 1.3: 3D Production Pipeline [Bea12]. Pre-production: Idea, Story, Storyboard, Animatic, Design; Production: Layout, R&D, Modeling, Texturing, Rigging/Setup, Animation, VFX, Lighting, Rendering; Post-production: Compositing, 2D VFX/Motion Graphics, Color Correction, Final Output.

scene, modeling the complex physical interactions remains challenging for vegetation. Vegetation has very abundant species and has different properties. Under the influence of external forces such as wind generated by passing object, different vegetation may exhibit various motion effects, ranging from oscillating lightly to swaying crazily.

When we are looking at an image, such as a photograph or a painting, usually the existence of dynamic or moving objects can be perceived, although the image is indeed static. A painting looks vivid because we may feel the presence or hints of motion in it. For instance, if we look at a picture of a tree, we may think of it swaying in the wind. This

FIGURE 1.4: Movie frames from animation “Twins on the Pasture” (©CCTV Animation Inc.). Wind is blowing, but the vegetation in foreground remains still.

motion effect helps us to better appreciate a photograph or a painting. On another note, nature scenes (especially vegetation scenes) exist in many cartoon animations. However, they still lack movement in some cases. For example, in the cartoon “Twins on the Pasture” (Figure 1.4), the flags are waving in wind, but the foreground grass remains still. Viewers may be unsatisfied with the absence of motion. For 2D vegetation animation, naive approach by manually drawing animation frames and interpolating them can produce vegetation animation, but it requires tedious effort and art skills. It has been observed that the movements of many types of vegetation (especially in cartoons) exhibit non-rigid property, such as tree crown, dense vegetation field, under water plant, etc. In their movements, the vegetation’s shape is perceived as deforming and oscillating, effects such as wave effect may also exists. However, existing methods of animating vegetation in images ([SATK99, CGZ⁺05]) mainly handle only tree type vegetation with branches whose motions usually exhibit rigid property. The existing 2D deformation methods generally adopt an as-rigid-as-possible approach [IMH05] which can not be directly applied to animate dense vegetation in 2D images, such as tree crown, bush field, grass field and etc. Moreover, existing methods mainly focus on modeling single elements such as grass blades, leaves and branches, instead of addressing explicitly the overall appearance of the

movements of a vegetation field, which we believe can be handled in a simpler and more efficient manner.

Although 2D computer animation is still widely used, 3D computer animation has become more common. Obviously, vegetation are an important component of outdoor landscape and animated 3D vegetation scene is required in a broad range of applications, such as virtual reality simulators, games, movies, and cartoons. 3D vegetation animation 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 (the rigging stage) to convert the “polygon soup” of vegetation is converted to a format suitable for 3D animation by setting up a 3D animation underlying model (bones, joints, etc.) ([[HKW09](#)] and [[DRBR09](#)]). Furthermore, existing 3D animation methods mainly focus on animating a 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 a high computational cost, for example, performing explicit finite element analysis for the tree mechanical model [[DRBR09](#)] and performing collision detection for blades of grass ([[WWZ+05](#)]) require high computational costs.

Modeling the motions of vegetation under the influence of natural wind has been widely studied. On top of that, designing and simulating specific motions of plants are required in many scenarios, especially in cartoons and animation movies where artistic (non-physically realistic) motions are required, such as the examples in Figures [1.5](#) (a) and (b, top). Short plants, such as grass, flowers, and shrubs, are common vegetation

FIGURE 1.5: Our method is suitable for scenarios which require designing and simulating specific motions of short plants, such as in cartoons (a), (b, top) and modeling the motions of short plants under the influence of a passing character (b, top) or car (b, bottom, from www.youtube.com). Cartoon frame (a) is from animation “Kung Fu Panda 2” (©DreamWorks Animation), (b, top) is from “Naruto” (©Masashi Kishimoto/Hayato Date/Shueisha/Studio Pierrot/TV Tokyo/Viz Media).

species and can be found in many scenes such as roadsides, grass fields, prairies, and gardens. Under the influence of external forces, short plants usually exhibit bending and twisting motions, such as grass bends and recovers when a car passes or a character pushes it. The motions of short plants reflect the character’s actions and the story, thus these specific motions should follow the animator’s design ideas. Furthermore, under the influence of an object, certain forms of pattern, pace, and trajectory are usually expected in the motions of short plants. For example, when a car passes, short plants may suddenly sway crazily and then gradually recover (Figure 1.5(b)). This is an important scenario in applications such as driving simulator and racing game. The short plants are expected to react according to the passing object’s motions. Efficiently authoring the motions of short plants is important:

1. For non-expert users, such as when teaching animation to primary pupils, it would be beneficial if they are able to intuitively and efficiently produce the motions of short plants that they observe in real-life such as a swaying grass in wind or they

imagine in mind, e.g., a dancing flower following the music.

2. For animators to conduct fast prototyping, it is important to rapidly realize or record their design ideas at the pre-production stage. Later at the production stage, the users can use other professional tools to further adjust the details based on the prototype.

In these applications, intuitively and efficiently realizing visually plausible motion effects of short plants is more important than the physical accuracy. Unfortunately, it can be difficult to achieve this using existing techniques. In the conventional animation pipeline, the rigging stage establishes a control structure (handles) for the 3D models of short plants, then the animation stage generates poses of the rigged models by computing the shape based on the transformations of the handles. These transformations of handles can be obtained from manual posing at many key frames, procedural methods or physical simulations.

Manual posing requires tedious work and art skills to move the handles through the traditional mouse and keyboard interfaces. Procedural methods can achieve visually plausible animations, however it is not always easy to design a procedure for a specific motion. Existing physics-based methods can generate physically realistic animations, however they usually require a high computational cost. For many applications such as real-time animation editing, there are usually many important computations need to be performed at the same time, such as rendering, character animation and fluid simulation, so it is helpful to lower the computational cost of plant animation while still making the animation looks visually plausible. Moreover, both the procedural and physics-based methods usually require setting a number of parameters. Additional knowledge and

tedious tuning are usually required to relate these abstract settings with the actual desired effects. Therefore, they are inconvenient and unintuitive. For instance, in the “Kung Fu Panda 2” (©DreamWorks Animation) example (Figure 1.5 (a)), it is difficult to generate the artistic motions using the procedural and physics-based methods. In the passing car example (Figure 1.5 (b)), it is in general a challenging task to produce the characteristics of the desired motions. Performing simulation of the wind generated by the car and its interactions with short plants as well as obtaining a number of properties (e.g., the mass of a blade) and tuning a number of parameters require tedious efforts. Another approach is by using motion capture data. However, in general, it is tedious to setup the motion capture system (e.g., markers and sensors) for short plants and process the captured data.

1.2 Objectives

This thesis aims at proposing new algorithms to address vegetation animations in both 2D images and 3D scenes. In more detail, the proposed algorithms have to fulfill the following requirements. This project aims to model common interactions and effects for a dynamic vegetation scene robustly at interactive frame rates, such as interactions with passing objects and swaying in natural wind. In the design of the methods, the properties of vegetation as well as the types of external forces are taken into account. Different types of vegetation react differently under external forces. For example, dense vegetation field usually exhibits soft or non-rigid property and short plants usually exhibit bending and twisting motions. Other than the properties of vegetation, the behaviors of vegetation also depend on how external force is applied to them. For example, shrubs vibrate lightly in breeze but sways crazily when a car passes them. Furthermore, motion propagation

is a common and essential effect for motions of vegetation: when a car passes a grass field, grass closer to the car bends first, then the bending propagates to other grass that are away from the car. The interactions with vegetation and the motion effects should be designed and controlled intuitively and in real-time by users. No special physics knowledge or art skill is required. The related work and state-of-the-art commercial tools (refer to the survey in **Chapter 2**) have been surveyed, and it can be concluded that there are rooms for research in the following directions:

1. Animating vegetation in 2D images

This work aims to animate vegetation in 2D images which can be hand-drawn images, paintings, cartoons or real photographs. This is helpful in enhancing the perception of photographs or paintings, creating wind visual effects in movies and assisting animators to create cartoon animations of vegetation. Vegetation scenes in many existing cartoon animations is still lack of movement. Therefore, **a method which can be used to animate vegetation efficiently in images is required.**

2. Animating 3D vegetation in real-time

This work aims to achieve real-time animation of 3D vegetation. The existing 3D vegetation animation methods still require tedious efforts to set up the underlying animation model and high computational cost. For real-time applications such as games, there are many tasks need to be performed such as AI and character animation, therefore, **it is important to lower the computational cost of vegetation animation while still making the animation look visually plausible.**

3. Interactive authoring of animations of short plants

This work aims to provide users with an intuitive interface for effective transition from user's targets or design ideas to actual animations. Instead of physical accuracy, our method focuses on creating essential visual effects of short plants which are perceived as either natural looking or artistic by viewers. Authoring specific motions of short plants is required in many scenarios, such as in cartoons. The conventional method requires tedious efforts and skills to set the poses. Performing physical or procedural simulation usually needs to obtain and tune many parameters. **An intuitive interface to rapidly author the essential motions of shorts plants is beneficial to non-expert users and conducting fast prototyping.**

1.3 Contributions

New algorithms are proposed in this thesis to animate 2D and 3D vegetation as further detailed in this section.

1. Animating vegetation in 2D images (Chapter 3, Papers 2 and 3 in Publications)

In general, for many applications mentioned above, the realization of the effects of vegetation is more important than the physical accuracy. Thus, instead of physics simulation, **a novel method has been proposed that combines a simple dynamics simulation and a procedural method to model the appearance of non-rigid movement of vegetation.** In our method, wind field is modeled using a fluid simulation. A harmonic oscillator based on wave simulation as the numerical model is adopted

for the dynamics of vegetation. The velocity generated from the wind field is used to drive the wave simulation to determine the velocity of vegetation. Based on the velocity of vegetation, a grid-based warping approach is employed to animate different types of vegetation such as grass and trees. Our method also considers the properties of vegetation such as the rigidity of the vegetation. Our method is simple and efficient, users can adjust the results to achieve their desired animation effects on the fly. As a result, using our method, users can interactively animate various types of vegetation in 2D images under different wind condition, ranging from vibrating in breeze to swaying in strong wind.

2. Animating 3D vegetation in real-time using a 2D approach (Chapter 4, Paper 1 in Publications)

Dimension reduction is one common problem-solving strategy in computer science. This inspires us to propose a new approach of generating 3D vegetation animation using a 2D approach. This work focuses on realizing the essential visual effects of natural-looking motions of a dense 3D vegetation field in real-time. This work does 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.

An efficient 2D approach has been proposed to animate 3D vegetation with densely leaved foliage under the influence of wind in real-time. In this approach, the 3D vegetation “polygon soup” is converted to view-dependent billboard layers and a 2D simulation of harmonic motion is performed at a simulation plane for modeling the dynamics of vegetation at the first layer (closest to the viewer). Then, the dynamics of the first layer is utilized to animate the rest of the vegetation layers while addressing the motion effects in depth and occlusion effects. This method does not require tedious operations to set up an underlying 3D animation model. This method does not perform an explicit 3D simulation. As a result, the computational cost can be reduced, which is favorable for real-time applications. 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.

The grid-based warping approach used in the proposed 2D and 3D animation methods was originally developed to synthesize 2D and 3D textures. Our texture synthesis method can be applied to model vegetation scenes such as synthesizing textures of grass and flower fields as well as textures of wood and bark for modeling tree trunks and branches. (**Paper 4 in Publications**)

3. Interactive authoring of bending and twisting motions of short plants using hand gestures (Chapter 4, Paper 5 in Publications)

Interaction technique is important in computer graphics. An intuitive and efficient interface plays a significant role in facilitating interactions and animation authoring. An intuitive interface has been proposed for rapid animation authoring for short plants based on the following observations and our real-life experiences:

1. Under the influence of external forces, short plants usually exhibit bending and twisting motions. Similarly, a hand can perform bending and rolling which can sufficiently represent the bending and twisting motions of a short plant (Figure 5.3). This is because mobility-wise, different from tall plants, short plants usually have lower degrees of freedom, which makes it possible for a hand (consists of wrist and fingers) to imitate the motions of a short plant.
2. Human is proficient in using a hand to mimic motions: when we are describing the motions of short plants to others, we naturally use hand gestures, for example, we usually use our bending hand to describe a bending grass blade (Figure 5.3).
3. In recent years, hand gesture based controller devices have become widely available, such as Leap Motion [Lea] and Nimble VR [Nim], which enable us to interact with a computer using our hands and fingers.

Many 3D hand gestural user interfaces have been developed to help us to perform our daily computer tasks such as switching between slides by swiping a hand as well as scrolling a page by moving a finger up and down. In these interfaces, a hand is used as a tool to trigger commands. Different from these interfaces, this work demonstrates that

a hand also can be used as a ‘puppet’ to model and mimic a motion. In this research, the acting capability of a hand has been employed. **A natural and intuitive interface has been proposed, facilitating rapid authoring of the animation effects that short plants usually exhibit: bending and twisting.** This approach transfers the motions of a hand to the motions of a plant element (grass blade or leaf). The global motions of the short plant are authored first, then the motions of its elements (such as leaves and flowers) are authored. A framework has also been proposed to utilize the animation results to animate a field of short plants and further adjust the motion effects according to the properties of the short plants, such as rigidity. This method is accessible to non-expert users and suitable for fast prototyping and authoring specific motions of short plants such as in cartoons.

1.4 Report organization

The remaining chapters of the report are organized as follows:

Previous work about the simulation of nature scenes and vegetation animation are discussed in Chapter 2. Chapters 3 and 4 describe the proposed algorithms to animate vegetation in 2D images and 3D scenes, respectively. Chapter 5 presents the proposed method to author animation for short plants. Conclusion and future work are given in Chapter 6.

Chapter 2

Related Work

If I have seen further, it is by standing on the shoulders
of Giants.

Sir Isaac Newton

In this chapter, we review the related work of 2D and 3D vegetation animations. We begin with a review of related work in modeling a nature scene (Section 2.1). Then, we discuss the related work of animation of vegetation in 2D images (Section 2.2) and 3D scenes (Section 2.3). Summaries are given at the end of Section 2.2 and Section 2.3 respectively.

2.1 Simulation of nature scene

A nature scene has many components, such as water, terrain, vegetation, and etc. The overall effects of a nature scene may be destroyed and users will not feel immersive, if we fail to model any one of them. Furthermore, there are similarities among the components of a nature scene and they are usually related to each other. The methods used in

modeling one component of a nature scene or one natural phenomena may be applied in modeling others. For example, these components of a nature scene or natural phenomena are both based on fluid simulation: Krištof et al. [KBKv09] simulated the erosion effect of terrain and Miyazaki et al. [MDN02] modeled cloud. Vegetation is an important component of a nature scene. Knowing essential methods of simulating various nature scenes and natural phenomena is helpful in developing algorithms to animate vegetation. Thus, in this section, we review some related work in modeling a nature scene.

2.1.1 Fluid

Fluid simulation is an important research area and basic tool in computer graphics for generating realistic animations of various natural phenomena such as fire smoke, cloud, water, and etc. Fluid is volumetric phenomena. Streams and rotational effects such as vortices and eddies are produced when each part of the volume interacting locally. This behavior can be captured using the well known Navier-Stokes equations. It describes how a velocity field changes over time while applying pressure and forces. Kajiya and Herzen [KH84] firstly introduced it to computer graphics. Foster and Metaxas were the earliest to apply the Navier-Stokes equations in full 3D [FM96]. Stam then developed a semi-Lagrangian advection technique to provide unconditionally stable behavior of fluid [Sta99]. This method enables much larger time steps and thus the simulation speed is increased. This technique was extended to handle complex 3D water simulations by Foster and Fedkiw [FF01] and by Enright et al. [EMF02] and Selle et al. [SFK⁺08]. Other most common methods are smoothed particle hydrodynamics (SPH) methods [MCG03, HKK07, SP09], vorticity-based methods [FSJ01], and Lattice Boltzmann methods [LWK03].

Smoke and fire are also fluid phenomena. Foster and Metaxas applied a solver of Navier-Stokes equations to approximate the motion of a hot, turbulent gas [FM97]. Fedkiw et al. [FSJ01] used the inviscid Euler equations which are generally more suitable for modeling gas. They reduce the computational costs of solving the Navier-Stokes equations. Nguyen et al. [NFJ02] proposed a physics-based method to simulate fire. They simulated thin flames by independently simulating both vaporized fuel and hot gaseous products using the incompressible Navier-Stokes equations. Zhang et al. [ZCM11] used a graph-based data-driven technique for fire modeling. An overview of fluid simulation in computer graphics is given in [Bri08].

2.1.2 Terrain

Fractal landscape modeling is one common approach to model terrain (refer to the surveys in [EMP⁺02] and [Dac06]). Stochastic subdivision methods are commonly used for fractal landscape modeling. To generate fractal terrain, Miller [Mil86] proposed a square-square subdivision method. Lewis [Lew87] used a generalized stochastic subdivision. Szeliski and Terzopoulos [ST89] proposed to combine deterministic splines and stochastic fractals.

FIGURE 2.1: Fluid simulation for wind [AK06].

FIGURE 2.2: Procedural method and wave simulator for wind [CJ10].

2.1.3 Wind

To fully and physically simulate wind, Akagi et al. [AK06] performed an explicit simulation of fluid dynamics by solving the Navier-Stokes equations (Figure 2.1), Pirk et al. performed the SPH simulation [PNH⁺14]. The cost of physical simulation is rather expensive. Some speed up calculations were applied to achieve real-time results. To avoid the fluid dynamics simulation, Shinya et al. [SF92] proposed a stochastic approach (FFT based) from wind engineering that modeled wind as a velocity field. To simulate wind effects for trees, Stam [Sta97] developed a similar stochastic process by performing a modal analysis and filtering random velocities in the frequency domain to establish the correlation among branches. In the game of Crysis [EA13] powered by the SpeedTree [Spe] engine, the wind animation system is entirely artist-controlled [Sou07]. The movement information of every single vertex on a plant was painted by hand which is tedious, however the results are impressive. We used a procedural method with a wave simulation to model wind effect [CJ10] (Figure 2.2). There are other methods to model wind, such as modeling wind directions based on noise functions [OFT⁺03, Zio07, HFC09, HCH12]

and wind projection based method [DRBR09].

2.2 Animation of vegetation in 2D images

The traditional animation format is 2D animation. It also can be referred as cel or hand-drawn animation [TJ84]. The traditional animation method is based on drawing each frame manually. Many 2D animation techniques have been developed to generate motion effects for 2D still images and cartoons. In this section, we review these prior works which are related to 2D vegetation animation.

2.2.1 Animating 2D still images

Shinya et al. [SATK99] proposed to couple physically-based simulation with skeleton-based morphing techniques to animate plants in 2D images. However only a brief idea was mentioned. Chuang et al. [CGZ⁺05] proposed to segment an image into layers and synthesize a motion texture to animate the still image layer by layer (Figure 2.3). Xu et al. [XWL⁺08] proposed a method to animate animals' motion in 2D images by finding the motion path and applying morphing technique. Schödl et al. [SSSE00] introduced video texture that can create repetitive and endless video based on a short video sequence [SSSE00]. Lin et al. [LWW⁺07] presented a system that generates high resolution animated videos from a small number of still images via the graph and Markov Chain model. These methods require a sufficient number of pairs of similar frames or images.

Example-based texture synthesis has been widely investigated, it can be applied to synthesize dynamic texture such as video texture and motion texture (refer to the survey

FIGURE 2.3: Segment a still image into layers and synthesize a motion texture to animate the image layer by layer. [CGZ⁺05].

in [WLKT09]). Wei et al. [WL00] proposed a pixel-based texture synthesis method to synthesize spatio-temporal textures, for example, smoke, fire, and water waves. Kwatra et al. [KSE⁺03] proposed a patch-based texture synthesis method based on graph-cut to synthesize texture (Figure 2.4). Kwatra et al. [KEBK05] proposed an optimization-based texture synthesis technique guided by flow to synthesize animated texture. Ma et al. [MWGZ09] combined example-based texture synthesis method and fluid simulation to synthesize motion texture to animate under water plant (Figure 2.5). Chen et al. [CKW⁺12] proposed a method to design a 2D time-varying vector fields which can be applied to animate still images.

FIGURE 2.4: Synthesis video texture using patch-based texture synthesis based on graph-cut. [KSE⁺03].

FIGURE 2.5: Synthesis motion texture incorporating fluid simulation to animate under water plant. [MWGZ09].

2.2.2 Simulation in cartoons

There are very little works in simulating plants in cartoons. Fiore et al. [DFVHVR03] proposed to build realistic 3D geometries of tree as underlying models to animate cartoon style tree (Figure 2.6). Haevre et al. [HFR06] applied their method of animating 3D tree to animate cartoon style tree. However, these methods depends on the explicit 3D models of trees.

FIGURE 2.6: Animate cartoon style tree based on building realistic 3D geometries of tree as underlying models. [DFVHVR03].

Hair animation in cartoons has some similarities with vegetation animation in cartoons. Eiji et al. proposed simulation [SY05] and data-driven [SKM⁺06] approaches. Noble et al. [NT04] used NURBS surfaces to model and animate cartoon hair. Shin et al. [SHM06] introduced a particle based method to render cartoon hair.

For other effects in cartoons, Liao et al. proposed procedural methods to simulate water [LYP11] and cracking [LY12] effects. Eiji et al. [SSTM09] also proposed a method for interactive shadow design.

In general, creating wind effect in cartoons requires a certain amount of art skills and tedious manual work [SY05], thus in most existing cartoon animations, the vegetation either remains still or moves periodically.

2.2.3 Summary of animation of vegetation in 2D images

Limitations of existing methods for 2D vegetation animation and possibility of improvements

Existing 2D animation methods usually focus on animating single elements such as branches, grass blades, leaves, and etc. However, **addressing explicitly the overall appearance of the movements is sufficient to achieve natural-looking animation for dense vegetation, especially for cartoon style vegetation.** The conventional approach would be to do simple 2D shape interpolation between key frames [Gen10], however it requires tedious effort and art skills. **There is a possibility that by performing a stable fluid simulation based on Navier-Stokes equations to**

automatically generate natural wind field [Sta99], we can create visually plausible wind effect without the requirement of tedious art works. Furthermore, existing methods mainly focus on animating tree-type vegetation and rigid movements, however, the motions of dense vegetation field usually exhibit non-rigid property, especially in cartoon style images. **Explicitly addressing the non-rigid motions can provide an efficient solution to dense vegetation animation.**

3D animation methods also can be applied to animate 2D vegetation. Fiore et al. [DFVHVR03] and Haevre et al. [HFR06]’s methods to add cartoon animation effects to trees are based on building 3D geometries of trees as underlying models. Then they perform animation in 3D and project it into a 2D cartoon. However, building 3D geometries requires tedious manual work, animation in 3D also requires a high computational cost. Moreover, a 2D cartoon style animation is different from an animation in 3D scene. Therefore, such 3D methods cannot be directly applied to animate vegetation in 2D images. More 3D animation methods will be discussed in the next section.

Limitations of existing methods for animating vegetation layer by layer

Existing 2D methods to animate 2D vegetation images layer by layer (such as [CGZ⁺05]) do not handle inter-layer interaction. Thus, the motion effects in depth, for example, when wind is blowing from front to back, the vegetation’s movements propagate from front to back, cannot be synthesized. Furthermore, their layers are static and they do not handle change of viewpoint.

FIGURE 2.7: Particle system [RB85].

2.3 Animation of vegetation in 3D scenes

Grass and trees are the most common and representative vegetation species in nature scenes. These two types of vegetation are in general very different. Grass is usually short and soft, on the other hand, trees are usually tall and complex. Other vegetation species such as flowers, shrubs, and underwater plants usually has similarities with either grass or trees. In this section, we review the 3D animation techniques for grass and trees respectively. Since the modeling methods usually affect the animation methods greatly, we also review the modeling methods for grass and trees.

2.3.1 Modeling of grass

Modeling of grass has been widely investigated. The major techniques to model a grass scene are particle system, billboards representation, volumetric representation, and 3D geometric representation.

Particle system

Reeves and Blau [RB85] used particles that move along parabolic trajectories to draw grass (Figure 2.7). The rendering cost is high, therefore it is not suitable for real-time

FIGURE 2.8: We [CJ10] proposed to use segmented billboards for grass modeling.

applications.

Billboards representation

Billboards have been widely used to efficiently render complex natural objects. Billboards are usually polygons textured using semi-transparent 2D textures. Billboards are the conventional way to represent grass. Habel et al. [HWJ07] and J. Orthmann et al. [JSK09] and the game Crysis [EA13] adopted the billboards to model grass. We [CJ10] proposed to use segmented billboards for grass modeling (Figure 2.8), the number of segmentation is set based on the height and softness of grass. In general, the billboards representation is simple and very efficient in rendering.

Volumetric representation

Kajiya and Kay introduced 3D texture mapping called "texel" to model furry surface [KK89]. Meyer and Neyret [MN98] extended this work by slicing 3D geometry sliced

FIGURE 2.9: Shell grass [BH02].

into a series of thin layers and incorporating the z-buffer techniques. A further extension called shell-based method used specifically for rendering fur and grass has been proposed by Lengyel et al. [LPFH01], Bakay and Heidrich [BH02] (Figure 2.9), Banisch and Wuthric [BW06]. The shell-based method is based on geometry layers mapped with semi-transparent textures and sliced parallel to the terrain. There are other work to render trees using volumetric representation. Pelzer [Pel04] proposed to model grass by crossing several polygons with the configuration that looks like stars. This configuration can ensure good visual quality independent of the current line of sight. In general, the volumetric representation is efficient in rendering.

3D geometric representation

Wang et al. [WWZ⁺05] used skeleton-lines to model grass blades. When the distance from the viewer increases, the number of skeleton-lines per blade decreases dynamically. There are some works using the level of detail scheme to combine 3D geometry representation, volumetric representation, and 2D texture of grass. Perbet and Cani [PC01] used chain of segmented polygons for 3D representation of grass and semi-transparent vertical textures

FIGURE 2.10: A grass field with three levels of details [BPB06].

of same orientation for volumetric representation of grass. Guerraz et al. [GPR⁺03] used many billboards for 3D representation of grass and semi-transparent vertical textures of different orientations for volumetric representation of grass. Boulanger et al. [BPB06] used a textured semi-transparent quadrilateral strips for 3D representation of grass and semi-transparent horizontal and vertical slices of textures for volumetric representation of grass (Figure 2.10).

Other methods

There are other ways to model grass such as parametric representation and L-System by Prusinkiewicz et al. [PPH⁺96] which included the botanical factors. Deussen et al. [DHBL⁺98] used lines and points to represent grass.

2.3.2 Animation of grass

There are several work for modeling grass-wind interaction. Perbet and Cani [PC01] proposed a procedural animation controlled by wind primitives. Bakay and Heidrich

FIGURE 2.11: Grass-wind interaction based on free form deformation and explicit grass-grass collision detection [WWZ⁺05].

[BH02] created an animation by translating the vertices of the shell grass according to a wind vector. Wang et al. [WWZ⁺05] combined the free form deformation and the explicit grass-grass collision detections (Figure 2.11). Endo et al. [EMF03] proposed a pre-computed physical model using dynamic constraints. Banisch and Wuthric [BW06] used a mass-spring system where the masses and springs are attached to the shell layers. Habel et al. [HWJ07] achieved a texture based animation by distorting the texture lookups. The conventional way to generate wind effects is by using the combination of sin and cos wave plus noise as in Pelzer [Pel04] and Zioma [Zio07]. Ramraj [Ram05] simulated the wind effects using a simple water wave. Spring simulation is also used to model grass animation [JSK09, FLHS15]. Nvidia released a grass tool: NVIDIA Turf Effects [Tre] to simulate and render massive grass simulations with physical interaction. AMD also released a tool: TressFX [Tre] to to simulate and render hair, fur, and grass.

Real-time continuum grass

In our previous work [CJ10], we proposed a method to simulate the interactions and natural effects of a large scale grass field in real-time for virtual reality and games (Figures

FIGURE 2.12: Continuum grass. We consider grass field as a two dimensional grid-based continuum and shift the complex interactions to the dynamics of continuum [CJ10].

2.2 and 2.12). Based on the grass in real-life, we focused on simulating the essential effects and behaviors of grass, such as bending and recovery. Grass should bend and recover in response to passing objects and wind, and the bending effect propagates to the neighboring grass. This propagation is mainly caused by grass-grass collision. Moreover, grass is hard to break and stretch but free to move in all directions. Other than the physical properties of grass, the behaviors of grass also depend on how objects and wind interact with them. For example, grass will likely bend according to the moving direction of the objects during the interaction. The grass properties as well as the objects and wind behaviors are considered in the design of our model.

Thus, we proposed a novel real-time method that combines a simple dynamic simulation and a procedural method instead of performing an explicit physics simulation. Our method to simulate grass integrates four types of interaction: grass-grass, grass-terrain, grass-object, and grass-wind. The proposed method has the following features:

(1) Grass field is treated as a 2D grid-based continuum, which stores the grass's status (bending angle and bending velocity). The flow inside the continuum is computed based on wave simulation. The propagation property of the wave represents the grass-grass interaction. We formulated a forward kinematics method to bend and recover the grass. The grass-terrain collision is tested during the bending process.

(2) We proposed a procedural method to handle the grass interactions with objects and wind. This method updates the continuum simulation (wave simulation) by directly setting the status of grass colliding with the objects and wind.

(3) Animators can control the simulation effects by setting the parameters of the wave propagation and the behaviors of the objects and wind.

(4) All computations of our method are performed in GPU. Our proposed method provides a simple and effective framework that enables a real-time animation of grass field. The simplicity and generality of our method allowed it to be a practical solution for real-time grass simulation.

2.3.3 Modeling of trees

Besides grass, trees are also an important member of the vegetation family. However, trees are different from grass in terms of material, geometric structure, and shape. Today, thousands of 3D tree models and libraries can be found on the Internet. To generate

FIGURE 2.13: Plastic trees: interactive self-adapting botanical tree modeling [PSK⁺12].

3D tree models, the conventional way is procedurally modeling using 3D modeling software such as 3ds Max Autodesk [Aut] or tree modeling software such as SpeedTree [Spe] and Xfrog [Xfr, LD99]. Achieving highly realistic tree models needs significant amount of efforts. Many methods have been proposed to simplify the modeling process, they can be categorized as below:

Rule-based method

The growth of trees follows botanic rules and environmental conditions. Rule-based methods are basically generating branches and leaves arrangement following a pattern which can be formulated based on a set of rules and parameters. Prusinkiewicz et al. [PJM94] proposed a series of methods based on L-system. In order to produce realistic-looking trees, Weber and Penn [WP95] proposed to use geometric rules. de Reffye et al. [dREF⁺88] designed rules based on botanical information. Pirk et al. [PSK⁺12] modeled trees incorporating environmental information (Figure 2.13). These methods can achieve good results, however they usually require efforts to tune and set the parameters and they also require expertise, such as botanical knowledge, for effective use.

Image-based method

Image-based method generate the 3D structure of a tree using images. Reche-Martinez et al. [RMMD04] proposed to generate a tree volume by recovering the opacity and color

FIGURE 2.14: Image-based tree modeling [TZW⁺07].

of each cell. Neubert et al. [NFD07] extract branch structure using particle flow. Tan et al. [TZW⁺07] proposed to generate the branches and leaves by computing a set of 3D points of a tree using structure from motion algorithm (Figure 2.14). These methods usually require multiple images as input and segmentation of multiple images.

Other methods

Tree models also can be generated based on 3D scan data [XGC07], however many efforts are required to scan a tree and process the data. Okabe et al. [OOI07] proposed to generate trees based on branch skeletons which are sketched by users. However, many user interventions and skills are required to sketch a realistic tree.

An overview of methods of representing trees is given in [BMG06]. Similar as grass modeling, billboards, volumetric techniques, and 3D geometric representation can be also applied to model trees:

Billboards representation

FIGURE 2.15: Billboard clouds tree model. [DDSD03].

Decoret et al. [DDSD03] introduced billboard clouds. They are basically an extreme simplification of a 3D tree model. To generate visually pleasing results, this method uses a set of arbitrarily oriented billboards to represent a geometry (Figure 2.15). To deal with plant models, many methods used billboards representation, such as Fuhrmann et al. [FUM05], Behrendt et al. [BCF⁺05], and Bao et al. [BZCJ09]. 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 proposed 3D vegetation animation method, we dynamically create view-dependent billboards of vegetation as the viewer moves.

Volumetric representation

Many components of nature scene are difficult to model using geometric primitives, such as fluids and clouds. Volumetric representation is usually used for modeling and rendering them. Similarly, volumetric representations are also commonly used for rendering trees. The most common model to represent a volume is a box with a stack of 2D slices.

FIGURE 2.16: Volumetric billboards [DN09].

Cohen et al. [CDN04] added realism to forest scenes encoded in volume by GPU-based lighting and shadowing. By pre-computing filtering textures and horizon maps, visually pleasing effects can be achieved. Decaudin and Neyret [DN04] proposed to use volumetric textures and aperiodic tiling. The forest is represented using a set of edge-compatible prisms which are aperiodically mapped onto the terrain. Decaudin et al. [DN09] improved their previous method by incorporating the GPU geometry shader and using volumetric billboards to present plant foliage (Figure 2.16). Volumetric representation is helpful to produce realistic rendering results. However, it is difficult to edit and animate volumetric models. Moreover, volumetric representation usually requires high memory cost.

3D geometric representation

The polygon is the primitive used for geometric representation of a tree. Level-of-detail (LOD) techniques have been widely used to eliminate of unimportant geometric details. LOD also provides the possibility to render and animate large-scale dynamic vegetation scenes in real-time. Deng et al. [DZGL07] presented a simplification method based on LOD. They proposed to represent nearby and far away coniferous leaves using cylinder and line respectively. Gumbau et al. [GCRR11] proposed a real-time vegetation rendering method using foliage pruning. Livny et al. [LPC⁺11] introduced a lobe-based representation to approximate the geometry of tree. Bruneton and Neyret [BN12] proposed to combine light field and height field ideas for rendering of the nearest trees as

FIGURE 2.17: LOD for realistic forest rendering [BN12].

well as for supporting the transitions of 3D models for close views. For rendering of the farthest trees, they used a texture generated using view and light dependent shaders (Figure 2.17).

2.3.4 Animation of trees

In general, the methods of tree animation can be classified into physics-based, procedural, and data-driven methods.

Physics-based method

The physics-based method usually considers a tree as a structural nodes embedded in a generalized coordinate system. The transformation of nodes is affected by its child or parent nodes. Treating joints as structural nodes, explicit or simplified integration of the equations of motion can be directly applied [SF92, SO99, GCF01, AK06, ZST⁺06]. However, it is expensive to apply integration for a high number of branches. Modal analysis and spectral method have been widely used to solve the complex dynamics of the tree. Modal analysis uses structural mechanics to determine the shapes and frequencies of a mechanical structure. Stam [Sta97] introduced modal analysis to animate a tree. The simulation is in the frequency domain and spectral method is used to reduce the computational cost. Chuang et al. [CGZ⁺05] also used a spectral method to model the

FIGURE 2.18: Physics guided animation of trees [HKW09].

animation of plants. They used a motion spectrum based on a damped harmonic oscillator. Habel et al. [HKW09] proposed a physics guided animation using beam deformation and a similar spectral method as [CGZ⁺05] (Figure 2.18). Diener et al. [DRBR09] proposed to compute a wind basis using modal analysis and project directional wind at run time (Figure 2.19). For model analysis, the most common geometrical representation for branches are beams. They have been described in [CGZ⁺05, HKW09, HCH12]. Moreover, Weber [Web08] proposed to use a cloth model for fast simulation of trees. Zhao and Barbic [ZB13] proposed to convert plant geometry to a format suitable for physics-based simulation, this method can be applied to animate many types of vegetation. Pirk et

FIGURE 2.19: Wind projection basis for animation of tree [DRBR09].

FIGURE 2.20: Procedural animation of trees in game Crysis [EA13].

al. [PNH⁺14] proposed to combine SPH simulation and botanical and environmental information of tree development to generate windy trees.

Procedural method

Procedural method is usually based on noise functions to heuristically model the appearance of tree motions instead of performing a physical simulation. For example, Ota et al. [OFT⁺03], Sousa [Sou07] (Figure 2.20), and Hu et al. [HFC09, HCH12] (Figure 2.21) used

FIGURE 2.21: Animation of interactive trees based on harmonic oscillator and noises [HCH12].

noise functions to model the animation. Procedural methods are commonly employed to generate the wind effect by using trigonometric functions or noise [Pel04, Zio07]. However, [AK06] performed an explicit fluid simulation to simulate the wind effect. [Ram05] simulated the wind effect using a simple water wave simulation. SpeedTree [Spe] 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 [Spe] 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.

Data-driven method

In data-driven method, the plants are animated based on pre-computed or pre-captured motion data. The major methods include: pre-computed motion graph [JTCW07, ZST⁺06, ZZJ⁺07], motion capture methods [LPJ15], and 2D video-based methods [SSSE00, DRF06, LDS⁺11].

Acceleration method Acceleration method includes LOD method [BK04, GCF01, ZST⁺06] and GPU based method [HKW09, DRBR09]. These methods speed up the computation of animation and provide the possibility to animate large scale vegetation scenes.

2.3.5 Hair and fur

There are similarities between hair, fur, and vegetation. The techniques such as mass-spring, free form deformation, and particle system have been used in simulating hair.

Bertails et al. [BHC⁺08] gave an overview of hair simulation. Hadap and Thalmann [HMT01] introduced a novel method to model dynamic hair as a continuum by performing a particle-based fluid simulation for lateral hair movement. Particles and hair strands are embedded in a fluid continuum. The hair-air, hair-body, and inter-hair interactions are handled using the fluid dynamics.

2.3.6 Summary of animation of vegetation in 3D scenes

Most existing methods are based on a combination of physics-based method and procedural method. They can be summarized as: performing fluid simulation or applying noise to create wind field, then directly applying motion equation or performing modal analysis to generate spectrum of motion (damped oscillator) to animate branches.

Limitations of existing methods for 3D vegetation animation and possibility of improvements

Most existing methods [SATK99] are based on setting up skeletons as the animation model for trees, so they are more suitable for tree type vegetation with branches, whose motions are usually rigid. Existing 3D methods [HKW09, HCH12, ZB13, PNH⁺14, Spe] 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 costs. Since the elements are dense and the elements at the back usually attract less attention and dense vegetation is usually non-rigid, **focusing on the apparent parts and essential effects such as the non-rigid property and the motion effects in**

depth are sufficient to achieve visually plausible animation. To reduce the computational costs, instead of directly animating the vegetation in 3D, we can perform the animation in 2D space and propagate the results to animate the whole 3D vegetation.

Limitations of existing methods for animation authoring for short plants and possibility of improvements

Existing physics-based methods [HKW09, ZB13, PNH⁺14] can achieve physically realistic animations, however they usually lack intuitive direct control and require a high computational cost. Procedural methods [OFT⁺03, Zio07] can achieve natural-looking animations, however it is in general difficult to design a procedure to achieve a given specific effect. Both the physical and procedural methods require to set and tune a number of parameters (some are not intuitive), unfortunately it is not always easy to relate these parameters to the actual effects. SpeedTree [Spe] can achieve visually realistic animations for many types of plants, ranging from grass to big trees. However, SpeedTree also requires the users to set and tune a number of parameters. Different from existing techniques, **we believe that the animation of short plants can be created in a more intuitive and efficient manner.** In data-driven methods, physical simulations or procedural methods need to be performed in order to pre-compute the data, such as constructing a motion graph [ZST⁺06]. Generating motion data from a 2D video may lack some 3D information [DRF06, LDS⁺11]. Motion capture plants in general require tedious effort to setup the system and process the captured data [LPJ15]. Such tedious efforts need to be reduced in order to increase the productivity in the animation pipeline. **An**

intuitive interface for rapid 3D animation can help users to easily and interactively generate their desired bending and twisting motions of short plants. Most existing methods only deal with the interaction between short plants and natural wind, however there is a need to author the motions of short plants interacting with objects, such as a passing car or a character.

Chapter 3

Animating Vegetation in 2D Images Using Simulation-guided Grid-based Warping

Until man duplicates a blade of grass, nature can laugh
at his so called scientific knowledge.

Thomas Edison

In this chapter, we propose a new method to animate vegetation in 2D images. We begin with an overview that describes the basic idea of our method (Section 3.1). We then address the important sub-problems, namely the segmentation and compositing (Section 3.2), wind model (Section 3.3), simulating dynamics of vegetation (Section 3.4), and animating vegetation in images (Section 3.5). Finally, we present results (Section 3.6), and end with summary (Section 3.7).

3.1 Basic idea

Given a single image with vegetation, which can be a hand-drawn image, painting, cartoon or real photograph, our target is to animate the vegetation in this 2D image. It can be applied to enhance the perception of a photograph or painting or to help animator to create cartoon animation of vegetation. We want to generate the following effects efficiently: (1) shrub oscillating in breeze, tree crown swaying in wind, under-water plant swaying in water, and wave effect of grass field, (2) when wind passing through a vegetation field, the plants closer to the wind source move first. The vegetation-vegetation interaction occurs. The movement is then propagated to other plants that are away from the wind source. To achieve the above two effects, we propose to combine fluid simulation, wave simulation, and image warping methods. Figure 3.1 shows the steps of the proposed method as follows:

1. We segment an input image into a vegetation layer and a background layer. (Section 3.2)
2. We perform wind simulation. The wind field is modeled using a velocity field. We perform a fluid simulation to generate a natural wind field, the solver is based on stable fluid [Sta99]. From this velocity field, we retrieve the wind's velocity vector, $\vec{V}:(V_x, V_y)$. (Section 3.3).

We choose to perform a wind simulation using fluid simulation (similar as in [AK06]) because fluid simulation can model the nature of wind: the flow of air. Furthermore, a wind field is actually a velocity field, performing a simulation is to generate

a velocity field based on the cause of the velocities (the input is force). Harmonic model like the one in [HKW09] can be considered as directly modeling a velocity field based on the characteristics of the velocities (the input is the characteristics of the velocities). We choose to perform a simulation, also because it can provide a higher level and intuitive control of wind such as blowing direction (flow direction) and wind strength.

3. We compute the dynamics of the vegetation, $\vec{V}_p: (V_{px}, V_{py})$. We adopt the wave simulation as the numerical model for the computation of dynamics of the vegetation. The velocity $\vec{V}$ from the wind field is used to initiate the amplitudes of the wave which represent the velocities of the vegetation $\vec{V}_p$. (Section 3.4).

We adopt the wave simulation to model harmonic motion effects. Harmonic motion is basically moving back and forth repetitively through equilibrium. Spring system is a simple harmonic motion example: when the system is displaced from its equilibrium position, a restoring force which follows the Hooke's law tends to restore the system to equilibrium. Wave is another harmonic motion example: the initial amplitudes of wave are set at some locations. The amplitudes start oscillating (moving back and forth) and propagating to nearby locations. Eventually, the wave will stop oscillating and reach an equilibrium position.

Essentially, many natural motions of vegetation can be viewed as harmonic motions. For example, under the influence of external forces like wind, a plant may sway back and forth (the movement can propagate to other nearby plants), when

the wind stops, the plant will eventually stop at a rest position. These effects of a vegetation field (swaying, oscillating) have similarities with the behaviors of wave.

Thus, we adopt the wave model and mimic those essential effects of a vegetation field using the wave simulation.

4. We animate vegetation layer using grid-based image warping method and the amount of warping is computed based on the vegetation's velocity $\vec{V}_p$. (Section 3.5).
5. The final step is the composition of the animated vegetation layer with background layer. (Section 3.2)

Our proposed method is suitable for animating dense vegetation scene in images (especially for cartoon style images), such as grass field, crown of tree, and group of bushes whose individual blades or leaves are dense and staggered together. Users can control the source, direction, and strength of the wind.

3.2 Segmentation and compositing

We can use existing methods such as [RKB04] to separate the vegetation to be animated (namely vegetation layers) from other parts of the input image, namely background. Another method is to segment the input image using magic wand tool of Adobe Photoshop [Ado]. Then, we perform image inpainting technique [EL99] to fill the empty area in the background. For every animated vegetation layer, image matting method [Ado] is employed to composite the vegetation layer with the background. In cartoon animation pipeline, animation frames are usually modeled in layers [Lay98]. In this case, we can integrate our method into this pipeline by applying it on the vegetation layers.

FIGURE 3.1: The steps.

3.3 Wind model

FIGURE 3.2: Wind field. The wind source is at the left boundary, and the wind direction is from left to right.

We implement a 2D fluid simulator to model wind (Figure 3.2). After one simulation time step, we perform vegetation animation and render one animation frame. The solver [Sta99] solves the Navier-Stokes equations:

$$\nabla \cdot \vec{V} = 0, \quad (3.1)$$

$$\frac{\partial \vec{V}}{\partial t} + (\vec{V} \cdot \nabla) \vec{V} = -\frac{1}{\rho} \nabla p + \vec{f}, \quad (3.2)$$

where $\vec{V}$ is the velocity, p is the pressure, $\vec{f}$ is the external force, ρ is the density, and t is the simulation time. Users can initiate wind by adding external forces to desired areas using a mouse. Users can control the frequency, direction, and strength of the external forces. Wind frequency is controlled by specifying the time interval of adding external forces during simulation. Users also can add external forces at any simulation time. External forces add velocities into desired areas. The velocities' directions and

magnitude are decided via setting parameters by the users. The velocities' magnitude is set between 0 to 1, which corresponds to the minimum and maximum strength of external forces. Then the velocities are updated by the external forces as determined by the 2D fluid simulator (Figure 3.2). The stable fluid solver ([Sta99]) ensures the magnitude of velocities is bounded by 1. We impose an open boundary condition, which means the velocity field is non-reflective at boundaries. This generated wind field is used to determine the vegetation's dynamics and guide the vegetation animation.

3.4 Simulating dynamics of vegetation

The velocity field generated using the fluid simulation, in general lacks oscillation effects which mimic the swaying motion of vegetation. Thus, it is not suitable to directly apply the result of fluid simulation as the dynamics of vegetation. We consider a vegetation layer as one continuum represented using a 2D grid, and assume the vegetation parts (such as leaves, blades) are evenly distributed in the continuum. We adopt the wave simulation to model the dynamics of continuum, and these vegetation parts are the simulation elements. We consider each simulation element's velocity $\vec{V}_p$ as the wave amplitude and apply the wave equation and then verlet integration to compute the simulation element's new velocity $\vec{V}_p$. The wave amplitude ($\vec{V}_p$) is initiated by the wind velocity ($\vec{V}$) which is obtained using bilinear sampling from the velocity field generated using the fluid simulator in Section 3.3.

The wave functions we used are the same as in [Zel04] and [CJ10]. For each simulation

time t , $a_{V_p}(t)$ is the accelerations of the wave propagation:

$$a_{V_p}(t) = \frac{\partial^2 \vec{V}_p}{\partial t^2} = c^2 \left(\frac{\partial^2 \vec{V}_p}{\partial x^2} + \frac{\partial^2 \vec{V}_p}{\partial y^2} \right), \quad (3.3)$$

$$\begin{aligned} \vec{V}_p(t + \Delta t) &= (1 + \gamma) \vec{V}_p(t) - \gamma \vec{V}_p(t - \Delta t) \\ &+ a_{V_p}(t)(\Delta t^2), \end{aligned} \quad (3.4)$$

By using the wave simulation, we provide a convenient way to control the vegetation animation. We can tune the wave damping factor, which reflects the softness of vegetation, by setting the γ (smaller γ : softer, greater γ : more rigid) and control how fast a vegetation propagates the wind effect by setting the wave speed factor c (smaller c : slow propagation, greater c : fast propagation). In our experiments, γ is in the range of $[0.01, 1]$ and c is in the range of $[1, 10]$.

3.5 2D vegetation animation

To generate the motion of vegetation, we define uniform grids over a vegetation layer and apply inverse warping to deform it. The grid's size s_x, s_y is defined based on the size of parts of vegetation, for example for vegetation with smaller leaves/blades, the size of grid is smaller; as well as the rigidity of the vegetation, for example for less rigid vegetation, the size of grid is smaller. The position of the four corners (x_0, y_0) , (x_1, y_1) , (x_2, y_2) , (x_3, y_3) of each grid cell over original vegetation layer is displaced (Figure 3.3). At each frame, we compute the amount of displacement $\vec{n}$ by Equation 3.5 using the vegetation's velocity $\vec{V}_p$ which is obtained by bilinear sampling from the wave simulation results in Section 3.3.

FIGURE 3.3: Grid-based warping. Left: Original vegetation layer with warped 2D grids. Right: Warped vegetation layer with 2D grids. Vegetation image is from www.colourbox.com (©Colourbox Educational).

A vegetation layer may have a perspective viewing angle, for example the vegetation image may be captured from top or side, thus we want to control the amount of deformation at a certain direction, for example, if grass is viewed from the side, it is unlikely for it to move up and down. We also want to control the amount of deformation at a certain region, for example, the vegetation far away should have less deformation compared to the near one. Thus we introduce a scale factor $\vec{\lambda}$ to control the amount of displacement $\vec{n}$. All of $\vec{V}_p$, $\vec{\lambda}$ and $\vec{n}$ consist horizontal (x) and vertical (y) components:

$$n_x = \lambda_x R V_{p_x}(t) \Delta t, \quad (3.5)$$

$$n_y = \lambda_y R V_{p_y}(t) \Delta t,$$

$$R = \begin{cases} 1 & \text{if every parts can move,} \\ (\frac{l}{h})^r & \text{otherwise,} \end{cases} \quad (3.6)$$

where Δt is the simulation time step, $\vec{\lambda}$ is set with respect to the grid cell's size, l is the distance to the horizon 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

FIGURE 3.4: Example of scale map. (a) Original vegetation scene. (b) Scale map, darker region means smaller magnitude for scale factor $\vec{\lambda}$. Vegetation image is from www.iwallscreen.com.

respect to the distance to the vegetation's horizon, whose position is fixed and determined by the users.

If the vegetation's root can be seen, the horizon position is set as the root part of the vegetation, which is usually the bottom of the vegetation layer (Figure 3.3). Greater r means the vegetation is more constrained by the root position so it exhibits less movement and is more rigid. If the vegetation's root cannot be seen, for example, vegetation field is viewed from top with a perspective angle, the horizon position is set at the top of the vegetation layer (Figure 3.4). As such, the vegetation further to the viewer exhibit much less movement than the vegetation closer to the viewer. h and l are computed by finding the distance to the bottom (Figure 3.3) or top (Figure 3.4).

Since our method can animate vegetation in real-time, the user can change the values of parameters on the fly to adjust the results. In the case that the vegetation field is very large and not homogeneous, we can segment it into different layers and animate each layer separately. However, this requires extra work for segmentation, thus we propose a simpler

control by defining a scale map (Figure 3.4). The scale map stores the magnitude of scale factor ($\vec{\lambda}$) for the vegetation layer, for example, the magnitude of $\vec{\lambda}$ of the vegetation region further to the viewer is smaller than the vegetation region closer to the viewer, which means the vegetation further to the viewer exhibit less movement. We directly set the magnitude of $\vec{\lambda}$ in Equations 3.5 and 3.6 based on the scale map in order to refine the velocity of the vegetation.

We use 2D grid-based inverse warping because it is simple, efficient, and GPU friendly [CJMW13]. It can well capture the deformation effects that a dense vegetation field exhibits. Using displacement $\vec{n}$, the four corners of grid cells in original vegetation layer are displaced in the opposite direction of the velocity, since our warping method is based on inverse warping. The new positions of the four corners after displacement (x'_0, y'_0) , (x'_1, y'_1) , (x'_2, y'_2) , (x'_3, y'_3) (Figure 3.3) are computed as follows.

$$(x', y') = (x, y) + (-n_x s_x, -n_y s_y), \quad (3.7)$$

For point (i, j) in the warped vegetation layer, we compute its corresponding point (i', j') in the original vegetation layer using bilinear interpolation based on its coordinates in grid cell (u, v) :

$$\begin{aligned} (i', j') &= (1 - v) \left((1 - u)(x'_0, y'_0) + u(x'_1, y'_1) \right) \\ &+ v \left((1 - u)(x'_2, y'_2) + u(x'_3, y'_3) \right). \end{aligned} \quad (3.8)$$

We sample the value at (i', j') at the original vegetation layer as the value of point (i, j) (Figure 3.3).

3.6 Results and discussion

FIGURE 3.5: Result animation using the proposed method using a cartoon (©Teemu Korhonen) as an input, the wind direction is from left to right. The most effected regions are highlighted in the second row.

The proposed method can be applied to animate vegetation in several types of input images, such as cartoon (Figures 3.5, 3.8, 3.12, and 3.13), photograph (Figure 3.6), and sketch (Figure 3.7). We applied our method to animate the types of vegetation that exhibit non-rigid property, such as dense vegetation: grass field (Figure 3.5), shrub field (Figures 3.6), tree crown (Figures 3.7, 3.12, and 3.15), and under water vegetation: sea wheat (Figure 3.8). We implemented our method on a laptop with Intel Core i7 M620 @2.67 Ghz CPU, 4 G Memory and Nvidia NVS3100M GPU on the DirectX 9.0c platform. With a 2D fluid simulator (64 by 64 grids) [Sta99], vegetation in images can be animated at 60 FPS. Please refer to the supplemental video in the accompanying DVD for the animation sequences.

Users can initiate wind by adding external forces to desired areas using a mouse and set parameters to control the frequency, direction and strength of the external forces. Users also can add external forces at any simulation time. The appearance of the vegetation animation can be controlled by specifying the grid's size (s_x, s_y) , the scale factor (λ_x, λ_y) , and the rigidity parameter (a) . Our method can animate vegetation in real-time, as a

FIGURE 3.6: Result animation using the proposed method using a photograph (©Moonglowlily) as an input, the wind direction is from right to left. (a) is one frame of the animated photograph sequence, for clarity, one part of the animated photograph sequence is enlarged and shown in (b), in (b) the most affected regions are highlighted in the second row.

result users can change the values of parameters on the fly to adjust the animation. We summarize the parameter settings and provide the setting guidelines as follows:

The grid's size s_x, s_y is defined based on the size of parts of vegetation (for example for vegetation with smaller leaves/blades, the size of grid is smaller) as well as the rigidity of the vegetation (for example, for less rigid vegetation, the size of grid is smaller). In Figure 3.12 the tree crown exhibits detailed motions and these motions are non-rigid, we choose to use a smaller grid size (5 pixels). On the other hand, the grass blades in Figure 3.13 are relatively larger and more rigid, we choose to use a larger grid size (10 pixels). Usually, the vegetation in real scenes exhibit more rigid motions while cartoonish motions are less rigid, for example in Figures 3.14 and 3.15, we choose to use a larger grid size (20 pixels).

Scale factor $\vec{\lambda}$ is set with respect to the grid cell's size, and it is set to refine the

FIGURE 3.7: Result animation using the proposed method using a sketch (©HUISJ.COM) as an input, the wind direction is from right to left. (a) is one frame of the animated sketch sequence, for clarity, one part of the animated sketch sequence is enlarged and shown in (b), in (b) the most affected regions are highlighted in the second row.

deformation at certain directions and locations. In Figure 3.12, the tree crown has both vertical and horizontal motions, therefore we use $\lambda_x = 1, \lambda_y = 1$. In Figure 3.13, the grass blades have more horizontal motions and less vertical motions, therefore we use $\lambda_x = 1, \lambda_y = 0.5$. The scale factor in the example in Figure 3.14 is set as the same as in Figure 3.12. The scale factor in the example in Figure 3.15 is set in the same manner as in Figure 3.4.

Rigidity parameter r ($[0, 10]$) controls the amount of displacement with respect to the distance to the vegetation's horizon, whose position is fixed and determined by the users. Greater r means the vegetation is more constrained by the root position, so it exhibits less movement and is more rigid. In Figure 3.12, every part of the tree crown can move, therefore we do not need to set r and just use $R = 1$ in Equation 3.6. In Figure 3.13, for the grass blade, the top of the grass can move while the root of the grass cannot, and the cartoon grass is relatively soft which means the top of the grass is less constrained by the

FIGURE 3.8: Result animation using the proposed method using a cartoon image from “The Adventures of Little Carp” (©CCTV Animation Inc.) as an input. The most effected regions are highlighted in the second row.

root, therefore we use $r = 0.1$. The rigidity parameters in the examples in Figures 3.14 and 3.15 are set as the same as in Figure 3.12 (every part can move).

3.6.1 Comparisons and analysis of results

We compare our result with real cartoon and video (Figures 3.12, 3.13, 3.14, and 3.15). The input is one frame of the cartoon or video. Please refer to the supplemental video in the accompanying DVD. In non-rigid motions for dense vegetation, the vegetation’s shape is perceived as deforming and oscillating, effects such as wave effect may also exists. Our method can capture these non-rigid motions for vegetation in images such as the shape deformation for grass in a cartoon animation (Figure 3.13) and the wave effect for tree crown in a video: the branches and leaves on the right bend first, then the bending

propagates from right to left (Figure 3.15). For the types of vegetation that exhibit non-rigid property, our method can achieve visually plausible result. Especially, for cartoons, such as animation sequences in Figures 3.12 and 3.13, our method achieves comparable results. For vegetation in real scenes (Figures 3.14 and 3.15), our method may perform less well.

Existing 2D vegetation animation methods (such as [CGZ⁺05]) usually focus on animating single elements such as grass blades and leaves. They can achieve detailed motions for vegetation elements such as a moving branch. However, our method explicitly models the overall appearance of the movements. Our method is suitable for animating cartoon vegetation images. Existing methods perform better when detailed motions are required, such as animating real vegetation images.

Furthermore, existing methods (such as [CGZ⁺05]) mainly focus on tree-type vegetation and rigid movements. However, our method focuses on dense vegetation and non-rigid motions which can be found in many cartoons. Our method cannot handle rigid motions, but can efficiently model non-rigid motions.

Existing methods (such as [SATK99]) need to construct the underlying animation model such as skeletons. It is usually not easy to robustly generate the skeletons. Our method is based on 2D grids, it is very efficient to define, and GPU friendly.

The conventional approach by creating and interpolating key frames [Gen10] requires tedious effort and art skills. Our method does not require tedious art works to create visually plausible wind effect.

We demonstrated that addressing explicitly the overall appearance of the non-rigid

motions is sufficient to achieve natural-looking animation for dense vegetation, especially for cartoon style vegetation.

3.6.2 Limitations

Our technique may perform less well for the type of vegetation that has obvious skeleton shapes such as branches. Because when a tree is swaying in wind, the shape of branches hardly changes, however, the shape of branches deforms in our method. Therefore, in our experiments, we segment branches as the background. Grid-based warping is simple but not sufficient to simulate the occlusion effect for vegetation under the influence of wind (for example, the leaves or blades may occlude each other). Thus, our method is more suitable to capture the overall appearance of vegetation's motion rather than the detail motion of parts of vegetation. For example, our method performs better for cartoon style images whose structure is simple (Figures 3.12 and 3.13), or vegetation (such as shrub) oscillating in light breeze, in these cases less non-occlusion artifacts can be noticed. For example, non-occlusion artifacts are more obvious in the real vegetation examples in Figures 3.14 and 3.15, than the cartoon vegetation examples in Figures 3.12 and 3.13. For some vegetation oscillating in light breeze, the vegetation's motion is perceived as each local part of it is slightly vibrating and the propagation and waving effects may not be obvious. Thus in this case, each local part may exhibit non-correlated motion and the wind direction may not be obvious.

Our technique may perform less well for the type of vegetation that has obvious skeleton shapes such as branches. Because when a tree is swaying in wind, the shape of branches hardly changes, however, the shape of branches is deformed in our method.

Therefore, in our experiments, we segment branches as the background. Grid-based warping is simple but not sufficient to simulate the occlusion effect for vegetation under the influence of wind (for example, the leaves or blades may occlude each other). Thus, our method is more suitable to capture the overall appearance of vegetation's motion rather than the detail motion of parts of vegetation. For example, our method performs better for cartoon style images whose structure is simple (Figures 3.12 and 3.13), or vegetation (such as shrub) oscillating in light breeze, in these cases less non-occlusion artifacts can be noticed. For example, non-occlusion artifacts are more obvious in the real vegetation examples in Figures 3.14 and 3.15, than the cartoon vegetation examples in Figures 3.12 and 3.13. For some vegetation oscillating in light breeze, the vegetation's motion is perceived as each local part of it is slightly vibrating and the propagation and waving effects may not be obvious. Thus in this case, each local part may exhibit non-correlated motion and the wind direction may not be obvious.

Our method involves wave simulation which models the propagation of movements. This propagation may be viewed as ripples. We can set the wave damping and speed factors to achieve desired propagation effects as discussed in Section 3.4. Our method is based on 2D grid-based image warping. During the warping, the image information in a grid cell can be magnified or minified. These will generate aliasing artifacts. The artifacts will be more obvious if the deformation amount is large with respect to the grid cell's size and if the input image is complex (e.g. large color variations). On the other hand, in our implementation, we only use a simple linear filtering and a simple full screen anti-aliasing. These methods are simple and fast, but they could not resolve such aliasing artifacts. However, in most cases, the deformation amount is not large and cartoonish

images (which are the main targets of our method) usually have less color variations. Therefore, we think these simple methods are sufficient.

In the future, we plan to explore other filtering methods to reduce the aliasing artifacts. For magnifying, we can reconstruct a continuous image based on the sampled input image, and apply the warping on this continuous image. For minifying, we can choose more adaptive anti-aliasing filtering methods, for example, construct a mip-map.

3.6.3 Generalization

Besides swaying vegetation, our proposed method can also be generalized to simulate other harmonic motions in 2D images like water surface perturbations and oscillations of elastic objects caused by fluid flow (Figure 3.9).

FIGURE 3.9: Parts of images exhibiting harmonic oscillations due to wind and fluid flow are highlighted: (a) trees and bushes, (b) water surface reflections, and (c) soft tissues inside the surgical scene being washed by the surgical fluid. The QR codes link to the actual videos published on YouTube.

To achieve smoother deformation results, instead of using bilinear interpolation in inverse warping (Equation 3.8), we propose to use bi-cubic interpolation as in [Bre]. As

FIGURE 3.10: Grid-based bi-cubic inverse image warping. Vegetation image is from www.colourbox.com (©Colourbox Educational).

shown in Figure 3.10, based on $\vec{n}$, we displace the corners of the grid cells $\vec{p}(x, y)$, ($0 \leq x, y \leq 3$), to the new position $\vec{p}'(x, y)$, ($0 \leq x, y \leq 3$). For one point (i, j) with coordinates (u, v) in one grid cell, we sample at the original input foreground layer at position (i', j') :

$$i' = \sum_{x=0}^3 \sum_{y=0}^3 a_i(x, y) u^x v^y, j' = \sum_{x=0}^3 \sum_{y=0}^3 a_j(x, y) u^x v^y \quad (3.9)$$

where $\vec{a}(x, y)$, ($0 \leq x, y \leq 3$) is the parameter for interpolation and it is computed with $\vec{p}(x, y)$ (refer to the Appendix A for its value).

Compared with bilinear interpolation (C0 continuity), bi-cubic interpolation (C2 continuity) can achieve a smoother deformation (Figure 3.11). In other words, using bi-cubic interpolation, the boundary of the deformed portion is more roundish. On the other hand, it requires a high computational cost. The deformation results of applying these

FIGURE 3.11: Comparing bilinear interpolation with bi-cubic interpolation. (a) Bilinear interpolation with small deformation amount. (b) Bi-cubic interpolation with small deformation amount. (c) Bilinear interpolation with large deformation amount. (d) Bi-cubic interpolation with large deformation amount. In the last row, the most affected regions of our results are highlighted.

two methods are similar. Only when the deformation amount is very large, the difference can be noticed. However, in our 2D vegetation animation method, very large deformation amount is usually avoided, since it may cause unwanted stretching artifacts. For the application of vegetation animation, it is not necessary to impose a deformation with high level smoothness. In the example in Figure 3.9 (c), to simulate the motions of soft tissues when the turbulent water flow is washing them, we choose to use the bi-cubic interpolation to achieve a smoother deformation.

This method can be integrated with haptic interaction to provide users a multisensory experience with images. As described in Sections 3.3, 3.4, and 3.5, we add harmonic motion effects to a still image. To realize multisensory experience, on top of the visual effects

we also want users to feel the forces which produce these motions. We can achieve this since the motion effects are generated based on flow (Section 3.3) and harmonic (Section 3.4) simulations. Essentially, these simulations generate the velocity of the elements ($\vec{V}_p$) in the foreground layer. Therefore, we can let the users feel these motions by converting the generated velocity field into the force feedback of the haptic device [CZ⁺14].

3.7 Summary

In this chapter, we have presented a new method for generating vegetation animation for an input 2D image. Our method does not require tedious work and art skills to create many key frames. This method addresses the non-rigid movements of vegetation in 2D images. Our method consists of the following features:

1. We consider the vegetation field as one continuum. The wind field generated using a fluid simulation, is used to drive the dynamics of the continuum.
2. Our method heuristically models the appearance of vegetation's motion. Many natural motions for vegetation such as oscillating in breeze or swaying in wind can be viewed as harmonic oscillation [CGZ⁺05, HKW09]. We adopt a wave simulation as the numerical model for the dynamics of vegetation.
3. We employ a grid-based image warping method to animate vegetation in images. The amount of warping is determined by the velocity of vegetation. Our method also takes into account the properties of vegetation such as the rigidity of the vegetation.
4. Our method is simple and efficient, users can adjust the results to achieve their

desired animation effects on the fly. Users can control the source, direction, and strength of the wind or water flow, and the property of vegetation.

As a result, various types of vegetation in 2D images under different wind condition can be interactively animated using our method, such as vibrating in breeze and swaying in strong wind. Especially our method is suitable for simulating dense vegetation in cartoons interacting with wind. For example, we can efficiently simulate grass, shrub, tree crown swaying in wind, and aquarium plants swaying in water flow.

FIGURE 3.12: Comparing our result with a real cartoon “Natsume Yuujin-Chou” (“The Natsume’s Book of Friends”) (©Yuki Midorikawa/Nihon Ad Systems Inc./Natsume Yuujin-Chou Project). The wind direction is from bottom right to up left. In the last column, the most affected regions of our result are highlighted.

FIGURE 3.13: Comparing our result with a real cartoon “Twins on the Pasture” (©CCTV Animation Inc.). The wind direction is from bottom right to up left. In the last column of each example, the most affected regions of our result are highlighted.

FIGURE 3.14: Comparing our result with a real video (from www.youtube.com). The wind direction is from right to left. In the last column, the wave effect for the grass field of our result is highlighted.

FIGURE 3.15: Comparing our result with a real video (from www.youtube.com). The wind direction is from bottom right to up left. In the last column of each example, the most affected regions of our result are highlighted.

Chapter 4

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

Nature has a great simplicity and, therefore, a great
beauty.

Richard Feynman

In this chapter, we propose an efficient method to animate vegetation in 3D scenes. Firstly, we describe the basic idea and steps of our method (Section 4.1). Then, we address the important sub-problems, namely creating vegetation layers (Section 4.2), 2D simulation of harmonic motion (Section 4.3), simulating dynamics of first vegetation layer (Section 4.4), guided dynamics of other vegetation layers (Section 4.5), and animating vegetation layers (Section 4.6). Finally, we present results (Section 4.7) and end with summary (Section 4.8).

FIGURE 4.1: Our method can be used to animate (from left to right) shrub, tree, underwater vegetation, and vegetation scene.

4.1 Basic idea and steps

Given a model 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 (Figure 4.1). 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.

The input 3D vegetation model can be a collection of polygons with no topology information. This unstructured representation model is also called “polygon soup” which is a common representation for 3D vegetation scenes with dense leaved foliage like tree crown and grass field which are composed of many thousands of leaves, flowers and grass blades. Note that one leaf or grass blade can be represented using one or several polygons. Our method does not rely on the type of input, it can handle 3D models such as 3D meshes (structured representation) and volumetric format. We present our method using “polygon soup” as the input models.

FIGURE 4.2: The steps. Red: input. Green: main loop. Blue: update. Dotted boxes: the corresponding sections in this chapter.

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 our proposed 2D vegetation animation in Chapter 3, which is a combination of a 2D fluid simulation and a 2D harmonic oscillator. We also extend the 2D grid-based image warping in Chapter 3 to simulate the motion effects in depth.

As shown in Figure 4.2, besides the 3D model (Figure 4.3 (a)), another input for our method is the simulation plane’s orientation, it defines the simulation coordinates (Figure 4.4). After generating vegetation layers (Section 4.2), we first perform the 2D simulation of harmonic motion at the simulation plane (Section 4.3). 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 4.4). This 2D simulation result 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 4.5). Based on the dynamics of vegetation, we animate the 3D vegetation model layer by layer (Section 4.6). Note that the simulation plane only needs to be defined once and remains static. The simulation results are projected based on the viewing directions.

4.2 Creating vegetation layers

We segment the vegetation’s 3D “polygon soup” into clusters parallel to the viewing plane, and each cluster of models is rendered into one vegetation layer, which forms one animation component of our system (Figure 4.3). 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).

FIGURE 4.3: Representation of a 3D shrub model. (a) The input 3D model. (b) Segment the model 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).

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 model. 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. We use multiple render targets: render to multiple frame buffer objects (FBOs), each FBO corresponds to one vegetation layer. Furthermore, the typical optimization techniques can be employed like instancing and mip-mapping. We also consider the temporal coherency, because the vegetation layers in current frame does not change too much comparing to the last frame. Thus, the vegetation layers may not need to be re-computed every frame.

The implementation depends on the application as well as the support of hardware and graphics API. If the application is bound by the bus between CPU and GPU, an alternative approach can be (1) pre-sort the leaves (polygons) in several camera directions and record the indexing information, fetching the indexing during runtime, (2) perform multi-pass rendering, set the clipping planes using graphics API and let the GPU to do the culling/clipping. Spatial partitioning methods such as octree can be employed to

accelerate the pre-sorting. They can also accelerate the vegetation layer computation via temporal coherency: only the leaves (polygons) at the neighboring layers need to be considered during the re-computing of vegetation layers. This approach requires more draw calls, however, less primitives are transferred from CPU to GPU.

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 4.7). 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.

4.3 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 4.4). 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 4.4). This dynamics is then used to approximate the dynamics

FIGURE 4.4: The wind field and nine tiles of the simulation plane.

of other vegetation layers (Section 4.5). Note that the simulation plane is static and the viewing plane can be dynamic: the position and orientation of the 2D simulation plane remain unchanged, only the simulation results are projected on the viewing plane which will be rotated as the viewing direction changes.

The 2D simulation of harmonic motion (based on our proposed method in Chapter 3) includes a 2D wind simulation (Section 3.3) and a harmonic oscillator (Section 3.4). 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 Chapter 3, Section 3.3 which applies the solver from [Sta99] 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 Chapter 3, Section 3.4. 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 Chapter 3, Section 3.4.

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 boundary conditions to create tilable simulation. In our method, we adopt a method, which is commonly used to make tilable textures as in [CJMW13], 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 [Ram05] can be applied and made tilable to model the dynamics of vegetation.

FIGURE 4.5: 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, and note that the simulation plane is static.

4.4 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 4.5 (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 4.5 and vertically shrinking the vegetation elements in Section 4.6.

When the vegetation layers are not parallel to the simulation plane, a projection is required to compute the dynamics of the first layer (Figure 4.5 (b)). (1) As in Figure 4.6, based on the simulation plane's position and orientation in world's coordinates, we

FIGURE 4.6: 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.

FIGURE 4.7: 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).

project the $\vec{V}_{s_{xy}}$, from the simulation coordinates to the world's coordinates $\vec{W}_{0_{xyz}}$:

$$W_{0x} = V_{sx}, \quad W_{0y} = V_{sy} \sin \theta, \quad W_{0z} = V_{sy} \cos \theta. \quad (4.1)$$

(2) As in Figure 4.7 (a), if the vegetation layers and the simulation plane has an angle α in XZ plane, $\vec{W}_0$'s Y component W_{0y} does not change and we decompose $\vec{W}_0$'s X and Z components in viewing coordinates:

$$V_{0x} = W_{0x} \cos \alpha + W_{0z} \sin \alpha, \quad V_{0y} = W_{0y}, \quad (4.2)$$

$$V_{0z} = W_{0x} \sin \alpha + W_{0z} \cos \alpha.$$

$\vec{V}_0$ 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 4.7 (b)) or an angle in XY plane in a similar way.

4.5 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 the 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 ($V_{0z} < 0$ 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 backtrack 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 $\{L_0 \dots L_{n-1}\}$ are the distances to the layer 0 for each vegetation layer, the velocity of vegetation at

time t is $\{\vec{V}_0(t)_A \dots \vec{V}_{n-1}(t)_A\}$:

$$\vec{V}_i(t)_A = \vec{V}_0(t - \gamma_i)_A, \quad (0 \leq i \leq n - 1), \quad (4.3)$$

$$\gamma_i = L_i / (-V_{0z}(t)_A + V_D), \quad (4.4)$$

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 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 4.3, 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 4.3.

4.5.1 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

FIGURE 4.8: 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 .

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 4.8, 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

$$\vec{V}_i(t)_A = \vec{V}_0(t - \gamma_i)_A \approx \vec{V}_0(t)_B, B = A + \gamma_i s \vec{V}_{0_{xy}}(t)_A, \quad (4.5)$$

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

FIGURE 4.9: The bending along Z direction appears as vertically shrinking.

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 4.3, 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,

$$-\frac{\vec{V}_{0xy}(t)_A}{\|\vec{V}_{0xy}(t)_A\|}.$$

4.6 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 in Chapter 3, Section 3.5. 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 4.3 and using the approximation method in Section 4.5. We extend the method in Chapter 3, Section 3.5 (Equation 3.5) to model the movements occurring along the direction perpendicular to the viewing plane (either towards or away from the viewer). As in Figure 4.9, 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} .

$$\begin{aligned} d_{i_x} &= R \left(1 - \left(\frac{L_i}{L_{n-1}} \right)^2 \right) m \lambda_x V_{i_x}(t) \Delta t, \\ d_{i_y} &= R \left(1 - \left(\frac{L_i}{L_{n-1}} \right)^2 \right) m (\lambda_y V_{i_y}(t) - \lambda_z |V_{i_z}(t)|) \Delta t, \end{aligned} \quad (4.6)$$

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. We use the same equation (Equation 3.6) in Chapter 3, Section 3.5 to compute R . Here, the horizon position is 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 $\left(1 - \left(\frac{L_i}{L_{n-1}} \right)^2 \right) m$, ($0 \leq i \leq n-1$) 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.

4.7 Results and discussion

We applied our proposed method to animate dense 3D vegetation: shrub, tree crown, grass field, underwater vegetation, and vegetation field (Figures 4.1, 4.10, 4.11, also please refer to the supplemental video in the accompanying DVD). Our method was implemented on a desktop PC with Intel Xeon E5520 2.27 Ghz CPU, 4 G Memory and Nvidia GeForce

FIGURE 4.10: Result shrub animation using the proposed method, the wind direction is from left to right. Our method can simulate the deforming and oscillating effects as well as the occlusion effects. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.

FIGURE 4.11: Result shrub animation using the proposed method. Our method can handle camera rotation and model the motion effects in depth. The wind direction was originally from left to right and it is now from left back to right front respecting to the current camera location and viewing direction. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.

GTX275 GPU. In our experiments, the 2D fluid simulator (with a 64 by 64 grid [Sta99]) 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 a better GPU than what we used in our experiments could further increase

FIGURE 4.12: Result tree animation using the proposed method, the wind direction is from left to right. Our method can simulate the deforming and oscillating effects as well as the occlusion effects. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.

TABLE 4.1: 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

the frame rate. With a frame resolution of 1024 x 768, we measure the FPS for two vegetation models/scenes with different number of faces and with different number of vegetation layers as shown in Table 4.1. 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.

FIGURE 4.13: Result underwater plant animation using the proposed method, the water flow direction is from left to right. Our method can simulate the deforming effects as well as the occlusion effects. (a) is one frame of the animated sequence, for clarity, one part of the animated sequence is enlarged and shown in (b), in (b) some of the most affected regions are highlighted in the second row.

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, as shown in Figures 4.10, 4.11, 4.12, and 4.13) 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 4.1) vegetation layers. For the last scene in Figure 4.1, 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.

4.7.1 Comparisons and analysis of results

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 [\[Spe\]](#) (please refer to the supplemental 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 [\[Swo\]](#) 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.

In general, short plants like shrubs and some rigid trees like pines, exhibit more rigid motions (vibrating: smaller movement amount, higher frequency), such as the examples in Figures 4.10, 4.11, and 4.15 as well as in Figures 4.16 and another pine example in the SpeedTree Cinema comparison results shown in the supplemental video. For tree crowns and underwater plants, they exhibit less rigid motions (swaying: larger movement amount, lower frequency) such as the examples in Figure 4.12, 4.13, 4.14, and 4.17. Besides, we presented many animation results in the supplemental video in the accompanying DVD. In the video, we also demonstrate that our method is able to handle different wind conditions: light wind and strong wind. Furthermore, we demonstrate a specific example showing the motion effects in depth: the wind is blowing from front to back and back to front with respect to the viewing direction. This is an important feature making our efficient 2D approach be able to handle 3D animation with a moving (rotating) camera.

Compared with the 3D methods such as [HKW09] and [Spe], their methods are able to generate detailed motions of each leaf and branch. Our method animates leaves in

groups. Therefore, for plants whose leaves are sparsely distributed and when the camera is very near to the plants, their methods perform better. However, for dense vegetation, it is very expensive to explicitly animate a large number (e.g. hundreds of thousands) of vegetation elements (leaves) in 3D. Moreover, for dense vegetation, many vegetation elements (leaves) usually stick together and move in groups, thus, it is not necessary to animate every element (leaf). Animation overheads can be reduced using our method.

Furthermore, for dense vegetation, the front parts of vegetation will largely occlude the parts behind and the front parts attract most attention from the viewer. Motion effects of vegetation at different layers have similarities. Based on these, we proposed to focus on the front layers of vegetation and applying approximation for the rest while accounting for the important effects: occlusion and motion effects in depth. This can further reduce the animation overheads. Our method addresses these characteristics of dense vegetation, therefore it is more suitable for dense vegetation animation in real-time applications.

For example, if we want to animate a dense tree crown with 4000 leaves, the explicit 3D method (such as [HKW09]) needs to animate all of them in 3D. Different from this method, our method focuses on animate the leaves which are closer to the viewer (maybe only 1000 leaves). We animate them based on 2D grids (e.g. a 20 x 20 grids), several leaves might correspond to one grid cell. As a result, the entities to be animated are significantly reduced and these grid cells only need to be animated in 2D. Moreover, the simulation is only required in 2D screen space.

Input wise, our method can handle “polygon soup” input which has no topology information. However, existing methods usually only can handle structured model. Based on our knowledge, there is no easy and robust method to construct an underlying animation structure from a “polygon soup” model. However, the 2D grids in our method can be

defined efficiently. Therefore, using our method, a lot of efforts to clean and pre-process the input 3D model as well as to construct an underlying animation structure (rigging, skinning) can be reduced. Implementation wise, our 2D grids model is very GPU friendly.

4.7.2 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 leaf 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.

As mentioned in Section 3.6.2, the propagation of movements due to wave simulation may be viewed as ripples. The ripple effects for single vegetation in 3D are less obvious than in the 2D case, since each layer contains relatively smaller number of vegetation elements than in the 2D case. For the grass field in the 3D vegetation scene example (the last scene in Figure 4.1), such effects are obvious because each layer contains more vegetation elements.

Since our 3D vegetation animation method is a 2D approach, it suffers from the same warping artifacts as described in Section 3.6.2. These artifacts may be accumulated. As a result, they are more apparent in the 3D case. Moreover, in general, since the “polygon soap” of dense vegetation has a large number of half-transparent polygons and they are densely inter-leaved, resolving the pixel coverage is not easy. More aliasing artifacts are generated in 3D case. Furthermore, our method is based on rendering to multiple vegetation layers and composing them, it resolves the pixel coverage within one layer, and then just composes the layers using depth information. The coverage resolving among layers is not handled, since the geometry information is not available during this composition. These may also generate some aliasing artifacts. The conventional multi-sampled anti-aliasing (MSAA) approach does not provide a good solution to it. The situation is similar as antialiasing is challenging in deferred shading (our method is based on rendering to multiple layers and it has a similar spirit as deferred shading). Currently, we use a full screen antialiasing (FSAA) as post processing to reduce the aliasing artifacts. In the future, we plan to explore other antialiasing methods, especially those antialiasing methods for deferred shading.

4.8 Summary

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

1. We convert the 3D vegetation “polygon soup” to view-dependent billboard layers

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). Our method does not require tedious operations to set up an underlying 3D animation model.

2. We utilize the dynamics of the first layer to animate the rest of the vegetation layers. 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.
3. 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.

Our method is suitable to simulate dense vegetation under the influence of wind. As a result, we can efficiently animate shrub, tree crown, grass swaying in wind as well as aquarium plant swaying in water flow in real-time.

Different from the proposed method of animating vegetation in 2D images, 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 our proposed 2D animation method. This 2D simulation addresses the essential animation effects of vegetation. The simplicity and efficiency of this simulation allow it to be a practical solution for simulating the dynamics of the first vegetation layer in this work.

FIGURE 4.14: Comparing our result with a real tree. In the last column, some deforming and occlusion effects are highlighted.

FIGURE 4.15: Comparing our result with a real shrub. In the last column, some deforming and occlusion effects are highlighted.

FIGURE 4.16: Comparing our result with a shrub animation created using SpeedTree Cinema (for movie applications). In the last column, some oscillation (vibration) and occlusion effects are highlighted.

FIGURE 4.17: Comparing our result with a tree animation created using SpeedTree Game (in a PC game *Sword Heroes' Fate 3* [Swo]). In the last column, some deformation (swaying) and occlusion effects are highlighted.

Chapter 5

Interactive Authoring of Bending and Twisting Motions of Short Plants Using Hand Gestures

You can design and create, and build the most
wonderful place in the world. But it takes people to
make the dream a reality.

Walt Disney

In this chapter, we propose an intuitive interface based on hand gestures for rapid animation authoring of bending and twisting motions for short plants. We begin with an overview that describes the basic idea and steps of our method (Section 5.1). Then, we discuss the related work of authoring motions of articulated object and hand gestures (Section 5.2). After this, we address the important sub-problems, namely the animation model (Section 5.3), animation authoring of a short plant using hand gestures (Section

FIGURE 5.1: Our method can be used to animate (first row, from left to right) grass, flower, shrub (two examples), (second row, from left to right) a scene of short plants (example of an ‘CG’ scene), and a scene of grass and short trees.

5.4), and using and controlling the authored animations (Section 5.5). A user study is given in Section 5.7. Finally, we present results (Section 5.6), and end with summary (Section 5.8).

5.1 Overview of the proposed system

Given a 3D model of a short plant, we aim to create animation effects (bending and twisting) that users desired: grass, flowers and shrubs swaying under the influence of the external forces such as a passing object or natural wind (Figure 5.1). We focus on

FIGURE 5.2: Transferring the motions of a hand to a short plant. The pitch and yaw motions of the wrist are transferred to the pitch and yaw motions of the root (joint 0) of the grass. They also determine the direction of bending for the upper parts (joints 1, 2, 3) of the grass. The roll motion of the wrist is transferred to the twisting of the upper parts. Bending angles of the finger joints control the magnitudes of the bending for the upper parts of the grass.

authoring these effects at the animation stage in the conventional animation pipeline. Similar to the global motions of a short plant, a hand can perform bending and rolling. Unlike tall plants, short plants have lower degrees of freedom, therefore using a hand can sufficiently represent the global motions of a short plant. Our authoring approach transfers the motions of a hand to the motions of a short plant (Figure 5.2). The transfer (not direct mapping due to fingers' limited degrees of freedom) is naturally derived based on how we describe the motions of a grass blade to others: we usually use our bending hand to mimic and describe a bending grass blade. Our method is for non-expert users to generate their desired motions of short plants as well as for professional animators to conduct fast prototyping. Users can use the proposed system for rapid animation creation during the design stage. They can use other professional tools to polish the resulted animations.

FIGURE 5.3: A hand can bend and roll which can represent the essential bending and twisting motions of short plants. Thus, it is natural to use a hand to author the motions of short plants. The followings are authoring examples of a grass blade. (a) Bending a hand forward and backward: grass bending forward and backward (pitch bending). (b) Bending a hand left and right: grass bending left and right (yaw bending). (c) Rolling a hand: twisting of grass.

We use the grass blade example in Figure 5.6 to introduce our basic idea. A wrist has three degrees of freedom (pitch, yaw and roll, refer to Figure 5.3), the pitch and yaw motions of the wrist are transferred to the pitch and yaw motions of the root (joint 0) of the grass, that means the grass can bend towards any directions. The roll motion of the wrist is transferred to the twisting of the upper parts (joints 1 . . . 3) of the grass, that means the upper parts of the grass can twist clockwise or counter-clockwise.

Note that the root and the upper parts of the grass, each have three degrees of freedom (pitch, yaw and roll). A straightforward mapping directly maps the wrist to the root and the fingers to the upper parts of the grass. This is not a good solution, since the fingers usually have only limited (one) degrees of freedom. In our system, all the joints of a finger have one degree of freedom (bend inwards to the palm and recover, refer to Figure 5.7). We propose to utilize this one degree of freedom: we let the bending angle of the fingers to control the magnitude of the bending for the upper parts of the grass. We make the direction of bending of the upper parts of the grass to be determined by the pitch and yaw motions of the wrist. This means the direction of the upper parts is the same as its root. This utilization enables the grass to bend to any directions. Moreover, connecting

root with its upper parts also enables the grass to smoothly bend from one direction to another which enhances the ease of use of the system. When designing our system, we also take into account the users' abilities to move their joints. As such, the motions are fully controllable by the users and various natural-looking motion effects of short plants can be achieved. Additionally, we propose to use root motions of the grass to guide a procedural animation method to automatically generate the motions of its upper parts. This is suitable for conducting a high-level authoring of the overall motions of the grass.

We can further generalize our model from a simple blade to a more complex short plant such as a flower (Figure 5.6), since it also has the same essential motion effects: bending and twisting. We treat the whole short plant (flower) as one element and author the global animation. We then work on the secondary elements (leaves) of the short plant and author the secondary animations. These two levels of animations are superimposed to create the final animation. Introducing secondary motions will add richness to the overall animation effects. Moreover, we propose this two-level approach to make the task in each step simpler. Furthermore, if users only want a simpler motion, they can choose to perform only one level authoring. We also introduce a method to align the secondary animations with the global motion.

Because different parts of a vegetation field have similarities, we propose to utilize the authored animation results of one short plant to animate the whole field. To realize the motion effects generated by external forces (such as passing objects or natural wind), we propose to use a motion propagation method base on backtracking in time. The tracking path is determined via an animation looping technique. Users can further control and adjust the motion effects according to the properties of the short plants. The backtracking

FIGURE 5.4: Our system setup.

in time is also used to model the rigidity of motions.

Our system (Figure 5.4) consists of a PC with a monitor and a hand tracking device from which we can get the information of a hand: the bending angles of the finger joints and wrist. In our implementation, we choose to use Leap Motion controller with its SDK.

As shown in Figure 5.5, we first set up the rig for the input 3D short plant model manually (Section 5.3). Then, during the animation stage, we perform the authoring of animation for one single short plant using hand gestures (Section 5.4). This authoring process is performed for the global animation first. This step may be repeated for the secondary elements (leaves). After authoring, the results are used to animate all the short plants in a field (Section 5.5).

FIGURE 5.5: The steps. Dotted boxes: the corresponding sections in the chapter. We first do the rigging for the input 3D short plant model manually (Section 5.3). Then, we perform the animation authoring for one single short plant using hand gestures (Section 5.4). This authoring process is performed for the global animation first. This process may be repeated for the secondary elements (leaves). After authoring, the results of one short plant are utilized to animate the whole field (Section 5.5).

5.2 Background of authoring motions of articulated object and hand gestures

5.2.1 Authoring and controlling motions of articulated objects

Animating an articulated model with skeletal structural is a time-consuming task [GGP⁺00]. Motion capture methods can be used to animate skeletal characters, however it usually requires motion re-targeting which includes complex computation [Gle98]. Computer puppetry methods [Stu98, SLSG01, SMTH13] can directly control and animate a character using a specific input device. Most recently, Jin et al. [JGGN15] proposed algorithms to map and transfer motions between skeletons. However existing methods mainly focus on character motions. For indirect mapping, it requires to select features and map them to another character rig with different structure. Hecker et al. [HRE⁺08] proposed to pre-record skeletal motions and map them to specific characters with different morphologies. Lockwood and Singh [LS12] used finger touches to generate motion data to control leg movements. Seol et al. [SOL13] proposed to puppet characters with a different topology. Coarse-to-fine design [MWT11, KCG⁺14] is a common approach to create animations. Dontcheva et al. [DYP03] proposed to use layered multiple passes of acting to create and edit character animations. Their layered acting is similar to our proposed framework, our framework can be considered as consisting of multiple layers. Different from their approach, we also focus on the interaction between layers, e.g., the root influences the upper parts. Rhodin et al. [RTK⁺14] also proposed a method to map different input motions to different character motions, such as mapping a moving human arm to a running horse. Oshita et al. [OOSM14] proposed to use hand motions to control

human character animations. The mappings from the captured motions to the character motions are not intuitive. In contrast to the existing methods, our method is based on the observation that it is natural to use hand gestures to describe the motions of short plants, hence we can transfer hand motions to plant motions while still maintaining the general intuitiveness.

5.2.2 Hand gestures

Hand motion capture devices have become widely available. Hand gestures are important interaction techniques. Please refer to [WWS⁺15] for a survey.

Motion capturing hand: The major methods to capture hand motions include: marker-based methods ([Vic]), glove-based methods ([Cyb]), and image and depth sensor based methods ([Lea]). **Applications of hand gestures:**

- (1) Triggering commands: Hand gestures can be used to triggering commands such as moving fingers to scrolling web pages or swiping a hand to switch through open browser tabs [Lea].
- (2) Manipulation: Hand gestures such as grasping can be used to performing tasks such as moving objects [KP06]. They can also be used to play instruments, such as a guitar [ES03], using complex positioning of fingers.
- (3) Communication: Hand gestures can be used for nonverbal communication and sign language [GMS⁺00].

Summary: The existing applications of hand motions are mainly for triggering commands and manipulating objects. In contrast to the existing applications, in this paper the acting capability of a hand is employed. We use a hand as a ‘puppet’ to act the

FIGURE 5.6: The underlying animation models for a grass blade and a flower. Their joints are labeled with respect to the joints of a hand. The yellow labels show the joints of the grass blade and the whole flower. The green labels show the joints of a secondary element (leaf) which is considered as a simple short plant.

desired motions of plants. Since human are proficient in using a hand, the motions of plants can be efficiently realized.

5.3 Animation model

As shown in Figure 5.6, for the underlying animation model, we use a skeletal structure to model a grass blade (simple short plant). For a plant with a more complex structure (a complex short plant, such as the flower in Figure 5.6), we consider it as one trunk and use a skeletal structure to model the whole short plant. Its secondary elements (leaves) are treated as simple short plants. Since the underlying model is very simple, we can easily set the skeleton manually using an interactive 3D modeling tool such as [Aut]. The

number of joints is set as the number of joints of the principle finger plus one for the wrist. In our implementation, we use a thumb as the principle finger (we will explain the reasons later in the next section), and it contains three joints. So in our skeleton articulated model, there are four joints in total.

For the 3D deformation, we adopt the most common linear blend skinning (LBS) method. The articulation parameters are affine transformations, each of them is associated with a bone as the handle. For a vertex v in the skin local space, its updated position v' in the world space is computed as:

$$v' = \left(\sum_{i=0}^{n-1} w_i M_i \right) v, \quad (5.1)$$

where M_i is the affine transformation matrix for joint i , and w_i ($i = 0, \dots, n-1$) represents how much influence the i -th joint has on vertex v . n is four in our implementation, 0 is the root (bottom) joint and 3 is the top joint. We define a simple linear method to assign the weights for the vertices, please refer to Appendix B. The weights are computed based on the distances from the vertex to the joints. Alternatively, the skinning weights also can be painted through an interactive 3D modeling tool such as [Aut](#).

5.4 Animation authoring of a short plant using hand gestures

We design the following authoring interface based on the transferring the motions from a hand to a short plant. We focus on the two essential motions of short plants: bending (Section [5.4.1](#)) and twisting (Section [5.4.2](#)). The additional semi-automatic method is introduced in the next section. An authoring is performed using one hand, while another

hand is also involved for triggering the keys to start and end the recording of the authoring. We record the rotational motion data $\vec{P}_i$ which includes pitch, yaw, and roll angles ($P_{pitch(i)}$, $P_{yaw(i)}$, and $P_{roll(i)}$) of the joints ($i = 0, \dots, 3$) of the short plant's skeleton. At each frame, we record the authored motion data $\vec{F} = \{\vec{P}_0, \dots, \vec{P}_3\}$. The recording is performed at a frame rate of 30 FPS. In Equations 5.2, 5.3, and 5.4, we introduce a scaling factor $\vec{s}_i$ ($s_{pitch(i)}$, $s_{yaw(i)}$, and $s_{roll(i)}$) for the joints ($i = 0, \dots, 3$). $\vec{s}_i$ is set smaller for a more rigid short plant. We will explain how to set these scaling factors in Section 5.4.4.

5.4.1 Bending

(1) **Root bending:** We author the root bending of the short plant using the wrist by simply bending or waving the hand. The root bending is actually the overall bending of the entire short plant. It is mapped as the wrist's pitch and yaw motions (Figures 5.3 (a) and (b)). The wrist can bend towards any directions, thus its motions are suitable to model the overall bending of the short plant, moreover it determines the bending direction of the entire short plant. The pitch and yaw motions of the root joint of the short plant's skeleton are set as:

$$P_{pitch(0)} = s_{pitch(0)} W_{pitch}, \quad P_{yaw(0)} = s_{yaw(0)} W_{yaw}, \quad (5.2)$$

where W is the rotational data of the wrist. The short plant can bend to any directions and recover as well as smoothly bend from one direction to another according to the wrist motion.

FIGURE 5.7: Bending the upper parts of a short plant.

(2) **Upper bending:** We author the bending of the upper parts of the short plant using the principle finger by simply bending the finger joints (Figure 5.7). The bending angles of the joints of the principle finger are used to determine the bending amount of the upper joints of the short plant's skeleton. The upper parts of a short plant usually bend in the same direction as the bottom part, and they are different only in the magnitude of the bending. Fingers can only easily bend inwards to the palm or recover to its rest position, performing other motions may not be feasible for fingers. However, this one degree of freedom is sufficient to control the bending amount for the upper parts of the short plant while following the root bending direction of the short plant. The pitch and yaw motions of the upper joints ($i = 1, \dots, 3$) of the short plant's skeleton are computed as:

$$P_{pitch(i)} = s_{pitch(i)}(P_{pitch(0)})G_i/G_{R_i}, \quad (5.3)$$

$$P_{yaw(i)} = s_{yaw(i)}P_{yaw(0)}G_i/G_{R_i},$$

where G_i is the bending angle of i -th joint of the principle finger, and G_{R_i} is its bending range. For short plants, the motions of the root and upper parts are in general coherent and linked. The pitch and yaw motions of the upper parts are affected by its root bending

(pitch and yaw) in terms of not only the direction but also the magnitude. Thus, we incorporate the wrist (root) bending to compute the bending of upper parts. Basically, we scale G_i/G_{R_i} , and the root bending can be considered as scaling factors. Intuitively, this means if the overall (root) motion is little, the added motion for upper parts is also little. This simple equation helps to make the following process efficient: in order to bend the upper parts of the short plant from one bending direction to another, the users can only perform one step: (1) bending the wrist to the target direction while the principle finger can be kept bent, instead of performing three steps: (1) recovering the principle finger to its rest position, (2) bending wrist to the target bending direction and (3) bending the principle finger to achieve the target bending amount.

5.4.2 Twisting

The twisting of the short plant is mapped to the wrist's roll rotation, for example in Figure 5.3(c), the blade of grass is aligned with the palm. The roll motion (P_{roll}) of the joints ($i = 0, \dots, 3$) of the short plant's skeleton is computed as:

$$P_{roll(i)} = \begin{cases} 0 & , i = 0, \dots, 3 - \tau, \\ s_{roll(0)} W_{roll} & , \text{otherwise.} \end{cases} \quad (5.4)$$

We set the number of twist-able joints (τ) based on the types of short plant, for example, a rigid short plant is difficult to twist, thus it has less number of twist-able joints. For short plants on the ground or the leaves, the root is usually not twist-able ($P_{roll(0)} = 0$).

In most of our experiments, we set $\tau = 2$.

FIGURE 5.8: Authoring (a) the global animation for the whole flower, then (b) the secondary animation for the leaf.

5.4.3 Secondary animation authoring

The authoring process is performed for the whole short plant. If the short plant contains only one element (e.g., one grass has one blade), the authoring is finished. However, if it contains some secondary elements such as leaves and the users would like to superimpose some secondary motions (Figure 5.8(b)), they can repeat the authoring step once or several times to create the secondary animation(s). The secondary animation results are used to animate all the leaves of this short plant. For the secondary animation results, we usually randomly assign them to the secondary elements (leaves) if they are all similar, however for short plants with different type of secondary elements, such as the petal and leaf in a flower, we author and assign the secondary animations respectively. During the authoring of the secondary animation, the animation result for the whole short plant can be viewed and looped (Section 5.5). This provides users the option to get on-the-fly feedback of the overall motion effects they are creating, e.g., users can instantly anticipate if the motions of leaves still look good when the flower bends to different directions. Moreover, this also provides users with a reference for animating the secondary elements,

as the secondary motions and global motions are usually correlated. For example, we observed that when a short plant bends to one direction, the secondary elements (leaves) also bend accordingly. By looking at the global motion, users can author the secondary motions while taking into account the global motion's properties such as the swaying frequency and direction. This is helpful to ensure the global and secondary motions are coherent.

5.4.4 Hand gesture capturing setup

In Leap Motion controller's setup, the hand's palm is facing the camera. The thumb's bending direction is almost perpendicular to the camera's viewing direction, however the other fingers' bending direction is parallel to the Leap Motion camera's viewing direction. Hence, the thumb's bending is captured more accurately than others' because the tracking of the thumb has less occlusion artifacts. Therefore, we choose to use the thumb as the principle finger.

Using the Leap Motion controller (with SDK), we are able to track the wrist's motions and the thumb joints' bending angles. However, our approach is device independent, for example if we use some glove (marker) based hand motion capture system, we can also choose the forefinger as the principle finger. That means any hand motion capture devices can be adopted, as long as the bending angles of the wrist and some fingers can be obtained.

People usually have different capabilities in bending the wrist and the fingers. Hence, when we set the scaling factor $\vec{s}_i$ in Equations 5.2, 5.3, and 5.4, we also consider the user's

FIGURE 5.9: Creating an animation sequence with infinite duration.

ability to bend his/her wrist and thumb. This means, we also set $\vec{s}_i$ for calibrating the bending of wrist and thumb in order to map the user's comfortable ranges of bending wrist and thumb joints (G_{R_i}) with his/her desired bending ranges of plant's joints. This step only needs to be done once prior to the authoring.

5.5 Using and controlling the authored animations

We fetch the recorded animation data and feed it into the underlying skeleton model to animate the short plant based on the LBS skinning method (Equation 5.1), and M_i is the rotation matrix and it is computed by compositing the rotation matrices that correspond to the recorded $P_{pitch(i)}$, $P_{yaw(i)}$, and $P_{roll(i)}$ bending. We also provide user with the options to create seamless infinite animation sequences (Section 5.5.1) and use the animation result to simulate a dynamic vegetation field (Section 5.5.2) as well as some high level controls such as speeding up or slowing down the animation (Section 5.5.3).

5.5.1 Animation looping

We propose a method similar to the idea of video texture [SSSE00] to animate short plants using the authored animations. As shown in Figure 5.9, we first select some

transition frames from the m frames of the recorded motion data $(\vec{F}_0, \dots, \vec{F}_{m-1})$. We then play the frames sequentially, if we encounter a transition frame (not the last one), we can randomly select one of the other transition frames to proceed or continue to proceed. If we encounter the last transition frame, we randomly select one of the other transition frames to proceed. As a result, we can achieve a seamless infinite animation while retaining the authored animation pattern.

To achieve this, we choose one frame as a transition pivot and use it to search for other transition frames. In our implementation, we use the frame at the back $(\vec{F}_{m-1-d})$ as the pivot. The recorded m frames are equally divided into λ regions, at each region we find p most similar frames comparing to the pivot. By doing so, the distribution of the transition frames can be more uniform. The comparison is performed based on the adjacent neighborhood with a weighted window size $(2d+1)$ and the Euclidean distance. At region r ($r = 0, \dots, \lambda-1$), the transition frames of indexes $c_{r_0}, \dots, c_{r_{p-1}}$, are computed as:

$$c_{r_k} = \arg \min_j \sum_k b_i \left\| \vec{F}_{j+i} - \vec{F}_{m-1-d+i} \right\|, \quad (5.5)$$

where b_i s are the binomial weights and $j = 0, \dots, \frac{m}{\lambda}$, $k = 0, \dots, p-1$. Small λ and small p (less transition frames) mean the looping effect is less fuzzy, in practice we set λ in the range of $[2, 5]$ and p in the range of $[1, 3]$. During transition if the two frames differ a lot, we use linear interpolation to insert some frames.

5.5.2 Motion propagation

Suppose we aim to create a dynamic field of short plants under the influence of external forces whose direction is from left to right. The external forces can be a wind generated

FIGURE 5.10: During time γ_B , the movement propagates from position A to B .

by a passing object, a natural wind blowing across the field, or a character pressing the short plants. We generate the field by defining 2D grids to model the ground and randomly selecting some grids and placing one short plant at a random location inside each selected grid. The short plants can be sorted and indexed via the 2D grids. To create the animation, we first author one animation of a single short plant. In general, the bending direction should be mostly towards right in order to be perceived as the direction of the external forces is rightward, however some oscillations as well as a little swaying towards other directions can be added. Animation looping method (Section 5.5.1) is then used to create an infinite animation. Since different parts of the field have similarities in terms of motions, we propose to use this animation result of a single plant for animating the short plants in the whole field. For a field with multiple types of short plants, we consider all the short plants of one type as one field, we author and animate each field respectively, then combine these fields by superimposition.

To initiate the motion propagation effect, for the left most short plant at each row (left boundary) of the field, we randomly select one frame from the authored data as the starting frame from which we play forward the recorded frames to animate the short plant. Note that, during the animation looping, the left most short plant determines the

frame sequences for its row, and the animation of the rest of the short plants of this row are derived based on it. When the direction of the external forces is from left to right, the short plant on the right reacts later than the short plant on the left, so we use the previous motion of the short plant on the left to model the current motion of the short plant on the right. We propose a method similar to the idea in continuum grass [CJ10]. In Figure 5.10, consider the movement ($\vec{F}$) propagates from position A to B during time γ_B , the motion of short plant ($\vec{F}_B$) at position B is actually the motion of short plant ($\vec{F}_A$) at position A at γ_B time ago.

$$\vec{F}(t)_B = \vec{F}(t - \gamma_B)_A, \quad (5.6)$$

$$\gamma_B = \|B - A\| / V_M, \quad (5.7)$$

where V_M is the propagation speed of the motion. Based on the time difference γ_B and the recording frame rate (30 FPS), we can easily compute the frame difference ($30\gamma_B$) and fetch the motion data based on the frame index. By propagating the rightward bending effects of the short plants from the left to the right of the field, a rightward turbulence or natural wind can be synthesized. The above approach can also be used to simulate a dynamic field under the influence of external forces in arbitrary directions.

5.5.3 Animation control

Control functionalities: We provide users with the following controls which are categorized with respect to the effects that they are related to:

(1) Direction of the external forces:

(a) Direction of motion propagation in a field: It is achieved by searching A in Equation 5.6 in the opposite direction of the external forces.

(b) Direction of bending: It is achieved by rotating the bending direction which is defined as (P_{pitch}, P_{yaw}) . This direction follows the direction of the external forces. If users want to introduce more twisting effects to incorporate the new external forces direction, they need to re-author the motions.

(2) Strength of the external forces:

(c) Speed of motion propagation in a field: For a field under the influence of stronger external forces, the motions propagate faster across the field than a field under the influence of lighter external forces. It is achieved by setting the V_M in Equation 5.7.

(d) Speed of motion: For a short plant under stronger external forces, its motion speed is usually higher than the short plant under lighter external forces. Speeding up and slowing down the motion speed are realized respectively by decreasing and increasing the time step to forward the animation frames which are originally captured in 30 FPS by default.

(e) Scale of motion: For a short plant under stronger external forces, its bending amount is usually larger than the short plant under lighter external forces. We also provide users the option to up or down scale the bending motions by setting $\vec{s}_i$ in Equations 5.2, 5.3, and 5.4.

(3) Rigidity of the motion: As shown in [CJ10], a soft short plant's motion usually exhibits the property that its upper parts are dragged by the bottom parts, in other

words, the upper parts react later than the bottom parts. By using the proposed hand gesture based method, we can already mimic and synthesis this effect, on top of that, we also propose a method to further adjust the rigidity of the motion based on backtracking in time. At time t , instead of using $\vec{P}_i(t)$, we use $\vec{P}_i(t - \delta_i)$, ($i = 1, \dots, 3$), where δ_i is the time delayed for each upper part, a soft short plant has larger δ_i . In implementation, this is realized by fetching motion data $\vec{F}(\{\vec{P}_0, \dots, \vec{P}_3\})$ from the recorded frames based on the delay time δ_i , this is actually the same mechanism as we handle γ_B in Equation 5.7.

Control interface: We propose a hand gestural interface for the above controls: users can swipe hand in arbitrary directions to the Leap Motion's camera to set the corresponding direction of external forces. We map the speed of swiping to the speed of motion propagation as well as the speed of motion and map the length of swiping to the scale of motion.

We compared this interface with keyboard control: users can control the direction of the external forces by pressing the left, right, up, and down keys as well as increasing or decreasing the strength of the external forces. In our experiments we observed that, in this case of animation control, using hand gestures maybe interesting and intuitive, but it has no advantages over using traditional keyboard in terms of efficiency and effectiveness. Therefore, in our framework we keep the hand gestural interface as one option for animation control. We believe in the near future, an environment with only hand gestural input (e.g., no keyboard) may emerge and this interface can be useful then.

Animation editing: We propose a simple approach to edit the authored animation. If users are not satisfied with a certain part of the animation, they can mark the start

and end frames of that part and re-author an animation sequence to replace it. To insert the newly authored animation, we use linear interpolation to blend the boundary frames.

5.5.4 Guided procedural animation

In addition, we also propose a semi-automatic approach to animate a short plant by using hand motions to guide a procedural animation method [CJ10]. During the recording stage, we only acquire the motions of wrist and use it to author the animation of the root as in Sections 5.4.1 and 5.4.2. We then propagate the motions of the root to the upper parts as in [CJ10]. At time t , the motions of the upper joints ($i = 1, \dots, 3$) of the short plant's skeleton are computed as:

$$\vec{P}_i(t) = \vec{P}_0(t - s_{g(i)} G_3 \sigma_i) \quad (5.8)$$

where σ_i is the time delayed for each upper part, a soft short plant has a larger σ_i . We handle it using the same mechanism as we handle γ_B in Equation 5.7. We also propose to use the thumb bending to control the rigidity of the short plant, by introducing $s_{g(i)} G_3$ in this equation, where $s_{g(i)}$ is a scaling factor and G_3 is the bending angle of the top joint of the thumb which is generally the most flexible joint of the thumb. This approach can be integrated with the proposed animation looping and motion propagation methods to animate a field of short plants. This approach only authors the overall motions of the short plants which are actually the root motions of the whole short plant, the motions are then propagated to its upper parts automatically. Thus, it is suitable when only a high-level design of overall motions of the short plant is required, such as when modeling a simple grass blade swaying in natural wind. It is not suitable to author motions for short plants require detailed design for upper parts, such as for flowers, the petals (top

parts) may require detailed control and design.

5.5.5 Alignment of secondary animation

Based on the observation that when the motions of the whole short plant (global motions) are severe, the motions of the secondary elements (leaves) are usually also severe at the same time (the opposite is usually not true). To model this effect, the secondary motions need to be aligned with the global motions. Users can author the secondary motions to achieve this. In many scenarios, the animated vegetation scene may contain a number of secondary animations and they maybe out of such alignment. Thus, we also propose an automatic method to implicitly align the secondary motions with the global motions to maintain the consistency. The overall effects of the authored secondary motions are still maintained. Please refer to the supplemental video in the accompanying DVD for the comparison of out-of-alignment and aligned results.

This motion alignment is achieved by aligning three important motion characteristics: motion direction, motion frequency, and motion amount. The motion direction can be aligned by adjusting the direction of secondary motions as introduced in Section 5.5.3 (1). In this section, we focus on the alignment of motion frequency and motion amount. Since one motion is in general perceived as one sequence of poses, at each time we process one subsequence of the authored motion data. Each subsequence corresponds to the same fixed time range. We usually use 2 seconds per subsequence (60 frames for 30 FPS). After one subsequence of the secondary motion data is processed to be aligned with its corresponding subsequence of global motion, it still corresponds to 2 seconds, but the number of frames within this subsequence may change (speed up or slow down). We

count another 2 seconds (60 frames) secondary motion data after the end of the processed subsequence as the next to process. The process path is determined by the animation looping (Section 5.5.1).

We first compute the total motion frequency and motion amount (bending amount) within the subsequence with index j respectively, for the global motions of the whole short plant $(f_G(j), A_G(j))$ and for the motions of the k -th secondary element $((f_{S_k}(j), A_{S_k}(j)))$. To compute the total motion frequency, we count the number of times that the bending amount of the short plant (or the secondary element) achieves a local maximum or minimum within the subsequence. We perform this counting for all parts (root and upper parts) of the short plant (or the secondary element), and sum the counting results as the total motion frequency. Similarly, to compute the total motion amount within the subsequence, we sum the bending amount (the amplitude of the bending) for all parts.

We then align the secondary motions with the global motions. The basic idea of the alignment is based on matching their total motion frequencies and amounts via scaling. By doing so, we can align the motions while still maintaining the overall effects of the authored secondary motions. For example, if the global motions are getting faster, the secondary motions are also getting faster, while still maintaining their trajectories. For the subsequence with index j , to align the motion frequencies, we speed up the motion of all parts of the k -th secondary element as mentioned in 2(d) in Section 5.5.3 based on $r \frac{f_G(j)}{f_{S_k}(j)}$, where r is a random scaling factor between 0.9 to 1.1. Similarly, to align the motion amounts, we scale the motion of all parts of the k -th secondary element as mentioned in 2(e) in Section 5.5.3 based on $r \frac{A_G(j)}{A_{S_k}(j)}$. This method also can be used to align the motions of different types of short plants in a vegetation field.

5.6 Results and discussions

We applied our method to animate 3D short plants: grass, flowers, shrubs and trees (Figure 5.1, also please refer to the supplemental video in the accompanying DVD).

Our method was implemented on a workstation with Intel Xeon E5-2650 2.6 GHz CPU, 16 G Memory and Nvidia Quadro K6000 GPU. All the operations for controlling an animation in Section 5.5.3 can be performed in real-time, users can adjust the animation on the fly. Users can author animation sequences with arbitrary lengths. In our experiments, the users only need to author 10 to 30 seconds animation for the global animation and each secondary animation. Although the users may author several times to achieve their desired results, each authoring session takes only 10 to 30 seconds.

5.6.1 Comparisons and analysis of results

We evaluate our method by comparing our results with (1) short plants in cartoons, (2) real short plants, and (3) the state-of-the-art award-winning toolkit SpeedTree [Spe] (please refer to the supplemental video in the accompanying DVD). Real and cartoon short plants exhibit bending and twisting motions. Our approach can effectively generate such motions while addressing the characteristics of the motions in the original videos (observations in real-life) and cartoons (design ideas). This is important in many scenarios such as teaching animation lessons to pupils and fast prototyping. This is not a trivial task using existing methods (such as SpeedTree), but since human are proficient in using a hand, such tasks can be efficiently done using our method.

For instance, in the animation movie “Kung Fu Panda 2” (©DreamWorks Animation), the shrub moves dramatically when “Master Shifu” is teaching Kung Fu and demonstrating “Inner Peace”. Such artistic motions cannot be easily achieved using the existing physics-based and procedural methods by setting and tuning a number of parameters.

Based on our knowledge, in the conventional 3D animation pipeline, during the animation stage, animators move the handles of the rigged 3D model to create poses. For this “Kung Fu Panda 2” example, in order to achieve such a nice artistic motion of a shrub interacting with a water drop which is driven by “Master Shifu”: receiving the water drop, making an interesting turn, dropping it to the pond and recovering, we believe it requires art skills and efforts. We also consulted with a professional artist. He told us that it may require 2-3 hours to create this. He told us that in general there are two approaches. The first one is based on carefully setting a specific rigging, as the animator moves the body of the shrub, other parts will follow. However, this rigging is not easy to set. The second approach is based on the normal rigging and key frames. It may require several steps and iterations to author the primary motions and secondary motions, during each step, the animator needs to set and tune the poses and the motion curves. These two approaches all require tedious efforts and art skills. However, using our method, a similar animation can be generated in 7 seconds. If the result is not satisfying the user can efficiently re-author or edit it. Based on the authored result, details can be added later during the production stage. Moreover, using our method, even non-expert users are able to generate desired animations of short plants.

In the passing car example, a natural-looking animation can be generated in 15 seconds, however the existing physics-based ([HKW09, ZB13]) and procedural methods

([OFT⁺03, Zio07]) would require tedious efforts to perform simulation and tuning the parameters.

SpeedTree is able to create realistic animations for a range of different types of plants. For short plants, we demonstrate that our method can achieve comparable results and our method provides a more intuitive and efficient approach. For example, in our experiment, authoring a grass animation (15 seconds) using SpeedTree took about 10 minutes, however using our method only took about 20 seconds.

5.7 User study and evaluation

Participants: The participants include 3 professional artists with average of 6 years experiences in game/animation industry and 7 researchers of visual computing (4 are from the same research center of the authors, 3 are from other labs). Out of 10 participants, 7 of them are familiar with 3D gestural interfaces (e.g., Leap Motion). In our study, the participants performed animation authoring tasks on a workstation with a Leap Motion controller.

Procedure: The study includes four sections: warm-up, target animation authoring, open animation authoring, results evaluation, and final interview.

(1) Warm-up: We first give participants an explanation of our proposed interface and the procedure of our user study. Then, they are asked to try pitch, yaw, and roll motions using hand. Meanwhile, the scaling parameters in Equations 2, 3, and 4 may be set based on their preferences. After they are familiar with all the functions of our system,

FIGURE 5.11: User feedback. (a) Feedback of the system. (b) Feedback of the animation results. Please refer to Appendix C for the questionnaire and detailed feedback.

we proceed to the next section. This section took around 5 minutes.

(2) Target animation authoring: After warm-up, the participants are asked to author an animation of a short plant based on a given 14 seconds reference video footage of a short plant. They are asked to try to reproduce the characteristics of the motions in the given video. It took 2 to 5 minutes for them to create their satisfied animation results.

(3) Free authoring: In this section, the participants are asked to use our system for free animation authoring for short plants.

(4) Animation result evaluation: In this section, we invite the participants to evaluate various animation results created using our system, including (a) each other's authoring results as well as comparison results with (b) real and (c) cartoon short plants as well as (d) SpeedTree.

System evaluation results: System evaluation results: Figure 5.11 (a) shows the feedback from the participants about our system. In general, all the participants are satisfied with our system. They think it is natural to use hand motions to represent the motions of short plants and our hand gestural interface is intuitive. They consider our system to be helpful for non-expert users to easily realize their desired motions. They think the

two-step approach is natural, since during each step they can only focus on one task (the average rating is 4.9, all the scores are from 1 to 5, 1 for bad and 5 for good). They also think the option of authoring the secondary motion while watching the global motion is helpful (the average rating is 4.2). For the 3 professional artists, they think our method is more efficient than existing 3D software for creating a specific animation for short plants. They feel our system is very helpful especially when conducting fast prototyping and in the early stage of design where the illustration and communication of ideas intensively occur. We can efficiently support this since our method provides a solution for rapid authoring of animations for short plants.

Animation evaluation results: The step (4) is a qualitative study to evaluate if our method can create motions effects that are perceived as natural looking by viewers (Figure 5.11 (b)). The participants think they can create visually plausible results as the reference motions (including each other's authoring results, the overall average rating is 4.1, the average rating is 4.0 evaluated by the three professional artists and 4.1 evaluated by the other users). Comparing with real and cartoon short plants, they think our animation results are visually plausible and the characteristics of the motions in the original cartoons and videos can be well captured, especially for the short plants in cartoons (the average rating, cartoon: 4.8, real: 4.3). They think our authored animation result is comparable with the animation result using SpeedTree (the average rating is 4.6).

Motion similarity: Moreover, we also perform a motion similarity experiment to evaluate the animation results. This experiment is to test if the important characteristics of the reference motions can be captured: the frequency, pace and amplitude change of the motion. We select one feature point of a short plant, e.g., we usually use the tip of the

FIGURE 5.12: Motion similarity. (a) Cartoon wheat from “The Old Mill” from Walt Disney’s “Silly Symphonies”. (b) Our result. (c) Motion similarity chart.

grass or leaf (Figure 5.12 (a, b)). We track the feature point and the root of the short plant in each frame for both the authored result and reference video. The tracking is done manually: at each frame we look for the feature point (P_x, P_y) and the root (R_x, R_y) and record their positions. Based on these two positions, we calculate the approximated bending angle at each frame using $\arctan(\frac{P_x - R_x}{P_y - R_y})$. Then, we normalize the bending angles and plot the curves of bending angles versus frames for both the authored result and reference video. As shown in Figure 5.12 (c), we put the two curves together in one chart and observed that their shapes are similar, e.g., both curves generally have 2 peaks (frequency), the distances between the peaks are similar (pace), both curves show small bending amplitudes followed by a big bending amplitude (amplitude change). This shows that our method is able to capture the important characteristics of the reference motions. Please refer to the video in the DVD for the chart of each example.

Our results show that using the proposed method to author bending and twisting motions, we are able to rapidly create the desired natural looking motions for many scenarios for short plants such as grass, flowers, and shrubs, under the influence of external forces, such as passing objects and natural wind.

5.7.1 Generalization and limitations

Generalization: The proposed approach is an efficient solution for authoring bending and twisting motions. It can be used to animate other objects which also exhibit such motions, such as fur, short hair and some body parts of virtual characters such as short tail. Our method can be integrated in to the traditional animation pipeline, for example in the pre-production stage, using our method can do fast prototyping and illustrate/exchange design ideas, and during the actual production, animators can start based our authored results.

Limitation: Our method may not achieve physically realistic results for real plants. However, it has the advantage of being fast. As such, users can efficiently perform re-authoring to improve the animation results. We also provide users with the tools to edit and fine tune the effects.

In our method, there are basically two types of parameters. The first type is to calibrate the hand gesture capturing device. This is usually necessary and only needs to be done once prior to the authoring. The second type of parameters aims at providing an option to control or to fine tune the animation results. This is usually optional. First, since our method is fast, users can always choose to re-author the animation if it is not satisfying. Second, if users think that only a small part is not satisfying, this type of parameters provides the tool to adjust it.

The Leap Motion controller used in our implementation can detect slight hand jitter. These detailed motions can be utilized to add richness to the final animation results, however, in some cases, such detailed motions may be considered as unnatural looking

(noises) especially for cartoonish motions. To reduce unwanted noises, we can apply a low-pass filter (or a smooth kernel) as in the conventional signal processing when processing the hand gestures. This low-pass filter is an anti-aliasing filter to generate smoother animation results.

This filter can be set on-the-fly during capturing and authoring. Since our method has the advantage of being fast and effective, users can efficiently perform re-authoring to achieve the desired smoothness for the animation results. The users can also choose to apply the low-pass filter as a post processing step, when performing the editing.

Currently we do not explicitly perform collision detection and response. As a result, short plants may intersect each other when collisions happen. However, in the generated animation, short plants are continuously bending and twisting and thus it is not obvious to see the intersections when they occur.

5.8 Summary

In this chapter, we have proposed a new method to use a hand as a ‘puppet’ to author animations of short plants. Because of the intuitiveness and efficiency of our method, it is suitable for non-expert users and helpful for fast prototyping and authoring specific motions of short plants. The proposed method is based on the observations: different from tall plants, short plants usually have lower degrees of freedom of bending and twisting motions, thus these motions of short plants can be well mimicked with the movements of a hand. In addition, it is natural to use hand gestures to describe the motions of short plants. Our method has the following features:

1. We propose a novel hand gestural interface, facilitating rapid authoring of the animation effects that short plants usually exhibit: bending and twisting. Our approach does not require tedious effort to set the poses of short plants at many frames and physical simulations which usually require high computational cost. Our authoring process usually takes only less than one minute.
2. Our approach transfers the motions of a hand to the motions of a plant element (grass blade or leaf). Thus, our approach allows users to create their desired animation of short plants by acting or mimicking the motions using a hand as well as easily control the characteristics of the motions on the fly. Our method is not simply a straightforward direct mapping. We consider the limited (one) degrees of freedom the finger joints usually have, the user's ability to bend his/her wrist and fingers, and the ease of use. It allows users to efficiently author short plants bending to any directions and recovering as well as smoothly bending from one direction to another.
3. We propose a two-level coarse-to-fine approach to author the animation for one short plant, e.g., users can first author the animation for the whole plant (e.g., one flower), then for its secondary elements (e.g., the flower's leaves). An implicit animation alignment method is also introduced to align the animations between secondary elements and whole plant.
4. We propose an efficient framework to reuse, refine, and control the authored animation results. Users only need to create the animation for one short plant, the results can then be applied to animate a field of short plants. To simulate a controllable

dynamic field, we propose to utilize the authored animation results by incorporating an animation looping technique and a motion propagation method. Users can further adjust the motion effects according to the properties of the short plants. Both the motion propagation and rigidity property adjustment are based on the idea of backtracking in time from our proposed 3D vegetation animation method.

5. We also propose a semi-automatic method by authoring only the overall (root) motions of the plant element by a hand and using them to guide a procedural method to automatically synthesize the motions of its other parts. This is an option suitable for users who only want a high-level design of the overall motions of the short plants, omitting the design and control of the detailed motions, such as modeling swaying grass in natural wind.

Our method is suitable to simulate one short plant or one field of short plants, such as grass, flowers, and shrubs, under the influence of external forces. Especially, it is helpful when simulating specific motions of short plants is required, such as in cartoons. In this research, we demonstrate that the animation of short plants can be authored in a simpler and more efficient manner. Our method has a similar target as a concurrent work [JGGN15], which is the easy accessibility to non-expert users. Moreover, it is helpful as a practical tool for users to illustrate their ideas and conduct fast prototyping. Most importantly, it opens the door of developing interactive hand gestural interfaces that is based on using a hand to perform acting jobs instead of issuing commands. In this research, we introduce this expressive design paradigm in the domain of vegetation animation.

Chapter 6

Conclusion and Future Work

Stay Hungry. Stay Foolish.

Steve Jobs

6.1 Conclusion

Modeling interactive nature scenes in real-time is increasingly demanded. It is important for movies, animations, environmental assessment in construction, and real-time applications such as games, outdoor training simulations, driving simulators, and flight simulators. Vegetation is one of the most important components of nature scenes. However, animating a vegetation scene in real-time is still challenging, because a vegetation scene usually has complex geometry and may consist of large quantity of elements (grass blades, leaves, and etc.). In general, computer animation can be categorized into 2D and 3D animation. The conventional methods of vegetation animation are manually drawing (e.g., 2D cartoons) and setting poses of plants at many key frames (e.g., 3D animation

movies). These methods require tedious efforts and art skills. Physics-based methods can achieve physically realistic animation, but they require high computational costs and lack intuitive direct control. In this thesis, novel algorithms have been proposed for real-time animation of 2D and 3D vegetation. First, a 2D method to simulate harmonic motions of vegetation in 2D images was proposed. Then, this 2D approach was extended to efficiently animate vegetation in 3D scenes. Lastly, an intuitive interface was proposed to easily and rapidly design and generate animation of short plants.

1. Animating vegetation in 2D images

Animating the vegetation in a still 2D image is helpful in enhancing the perception of a photograph or painting. It would be interesting to interactively add wind effects to a 2D image to produce an attractive animation. It is also helpful to assist animators to create cartoon animation of vegetation. **A real-time method was proposed for effectively simulating wind effects for vegetation in 2D images.** A fluid simulation was adopted as the wind model and image warping for animating vegetation. This method does not need tedious efforts and art skills to manually draw many key frames. This method explicitly models the non-rigid movements of vegetation in 2D images, it is suitable for animating dense vegetation, especially in cartoon style images. As a result, for vegetation in images, this method can simulate grass and trees swaying in wind as well as under water plants swaying in water. Users can interactively control the location of the source, direction and strength of the wind or water flow and the property of vegetation on the fly. (Papers 2 and 3 in Publications)

2. Animating 3D vegetation in real-time

For real-time 3D applications such as games, it is important to lower the computational cost of vegetation animation while still making the animation looks visually plausible. Therefore, **an efficient 2D-based approach was proposed for real-time animation of vegetation in 3D scenes.** The vegetation meshes were segmented into view-dependent 2D billboard layers. The proposed 2D harmonic motion simulation was adopted for modeling the dynamics of vegetation at the first layer. A procedural method was also proposed to project the simulation result when the vegetation layers do not parallel to the simulation plane. Then, the vegetation's dynamics at the first layer was utilized to guide the dynamics of the other layers. The motion effects in depth and occlusion effects were also be considered. This method is suitable for simulating wind effects on 3D vegetation fields with densely leaved foliage. As a result, this method can produce natural-looking motions of vegetation swaying in wind comparable with existing commercial software. However this method does not require tedious work to set up the underlying animation structure or performing an explicit 3D simulation. (Paper 1 in Publications)

3. Interactive authoring of animations of short plants

Short plants, such as grass, flowers, and shrubs, are common vegetation species. It is still difficult to intuitively and efficiently author the motions of short plants using existing techniques. **An intuitive interface based on hand gestures was proposed for rapid animation authoring of bending and twisting motions for short plants.** The proposed method is based on the observations: different from tall plants, short plants usually have lower degrees of freedom of bending and twisting motions, thus these motions

of short plants can be well mimicked with the movements of a hand. In addition, it is natural to use hand gestures to describe the motions of short plants. This method is based on transferring the motions of a hand to the motions of a short plant. Since human are very proficient in using hands and fingers, the essential motions of short plants can be efficiently realized. Moreover, the properties of these motions can be effectively and easily controlled using a hand, such as the bending directions, frequencies, and trajectories. Therefore the characteristics of the targeted motions can be well addressed. To author the animation for one short plant, a coarse-to-fine authoring approach was employed. An implicit animation alignment method was also introduced. A semi-automatic method was also proposed, which is suitable for users who only want a high-level design of the overall motions of short plants.

An efficient framework was proposed to reuse the authored animation results of one single short plant to animate a field of short plants. As a result, users can intuitively generate their desired motions of short plants under the influence of external forces. This method does not need the tedious manual posing or the expensive physical simulation. This authoring process usually takes only less than one minute. Because of the intuitiveness and efficiency of this method, it is suitable for non-expert users and helpful for fast prototyping and authoring specific motions of short plants. (Paper 5 in Publications)

In conclusion, using the proposed 2D, 3D, and authoring methods for vegetation animation, the manual efforts of animators and the computational costs can be significantly reduced, and natural-looking vegetation animation can be intuitively and efficiently achieved for many types of vegetation such as shrub, flower, grass field, tree crown, short tree, and underwater vegetation. All of these three methods considered the properties

of vegetation such as the rigidity of vegetation. The proposed methods addressed the essential and common interactions and motion effects of vegetation under the influence of external forces such as natural wind and a passing object, in particular, they all modeled the essential motion propagation effects.

The proposed 2D method can be integrated into the proposed 3D animation framework. Both these two methods addressed the non-rigid motion effects that dense vegetation usually exhibit, such as oscillating and deforming. They demonstrated that for dense vegetation field, focusing on the overall appearances and the apparent parts of vegetation's motions can adequately achieve natural-looking animation, without tedious skeleton-based setting up and computation.

The 2D grid-based image warping technique used in the proposed 2D and 3D animation methods was originally developed to synthesize textures. The proposed texture synthesis method can be applied in vegetation scene modeling such as synthesizing grass and flower field as well as wood and bark textures. (Paper 4 in Publications)

The proposed authoring method is suitable for modeling the bending and twisting motions of short plants. It extended the idea of motion propagation (backtrack in time) from the proposed 3D vegetation animation method to not only simulate the dynamics for a field of short plants but also to model the rigidity of short plants. It is natural and proficient to use hand gestures to describe the motions of short plants, hence hand motions can be efficiently transferred to plant motions while still maintaining the general intuitiveness. Most importantly, it starts a new direction of hand gestural interfaces development which is based on using a hand to perform acting jobs instead of issuing commands.

Comparing our proposed methods with the state-of-the-art

SpeedTree [Spe] is the state-of-the-art tool to animate plants. SpeedTree was awarded a Scientific and Technical Academy Award. Our 3D vegetation animation method and short plants animation authoring method have been compared with SpeedTree. Both our methods can produce natural looking motions of vegetation comparable with SpeedTree. In our 3D vegetation animation method, the effort to set up the underlying animation model and the computational costs can be significantly reduced. In our animation authoring method, the efforts to set and tune a number of parameters can be reduced.

In summary, all the objectives of this research have been met. All the proposed methods are real-time. We believe they are good practical solutions for vegetation animation in real-time applications.

6.2 Future work

In the future, we want to further enhance the vegetation animation quality and animate more types of vegetation. We also plan to expand the proposed algorithms to solve other problems.

6.2.1 Animating vegetation in 2D images

Our proposed 2D vegetation animation method incorporated grid-based image warping to simulate dense vegetation field which exhibits non-rigid property. However, the movements of other types of vegetation usually exhibit rigid property: under external force

such as wind, tree branches tend to bend and recover rigidly. We want to explore other types of 2D deformation methods besides grid-based method to handle more types of vegetation and effects. The possible solutions include as-rigid-as-possible shape manipulation in [IMH05] and real-time deformation method in [JBPS11]. However, these methods require a certain amount of efforts to set up the underlying animation structure (skeleton, handles). We plan to automatically generate such handles by analyzing the vegetation image. For example, the skeleton of a branch in an image can be generated using the morphological image processing method.

6.2.2 Animating 3D vegetation

In Chapter 4, we have presented a 2D approach to animate 3D vegetation in real-time. We want to improve the current shading and incorporate ambient occlusion effects. We also plan to take into account human perceptual factors in the billboard layers generation, for example, focusing on vegetation elements that are more attractive, e.g., vivid in color.

Our proposed hand gestural interface is suitable for authoring animations of short plants. We want to extend this method to animate tall and complex plants by authoring one part of the plant and propagating the authoring to the whole plant. In our method, we proposed to use hand motions to guide a procedural animation. However, the motions of tall and complex plants usually are more complicated, thus it is not easy to define a procedural method to simulate complicated motions. Instead, we consider to use physics-based simulation. For example, we can use hand to author the overall motion effects of one part and perform physics-based simulation to generate the detailed motion effects. Rigging in practice often requires many manual efforts, therefore we plan to incorporate

the automatic skeletal rigging method in [LD14] to further reduce the time and efforts to author 3D vegetation animation.

6.2.3 Applications and generalizations of the proposed methods

We also want to animate other natural elements in images as well as in 3D scenes using 2D billboard layers, such as water, cloud, smoke, and fire. For example, the motion effects of cloud also exhibit non-rigid property, therefore our proposed methods can be applied to animate cloud. We also plan to explore using a hand to author other objects which also exhibit bending and twisting motions, such as fur and short hair.

In addition, it is possible to combine our proposed 2D animation method with tour into the picture [HAA97] to provide users with an even richer experience. To combine with the tour into the picture, we can also use our algorithms to handle the movement of camera from the proposed 3D vegetation animation method.

Our proposed hand gestural method is a new expressive modeling approach. We introduced this approach in the context of short plant animation. This is a first attempt to use hand to act plant motions. In the future, we plan to generalize our approach by considering different input devices, since currently gesture capture devices became widely available, for example, consumer standard gesture capture devices (Leap Motion and Microsoft Kinect), are able to achieve good gesture capture results. With the rapid development in Virtual Reality (VR), the conventional keyboard and mouse input interface might not be suitable. Gestural based input can be helpful for VR applications, for example, Leap Motion controller can be attached to a VR device to capture hand gestures

which are used to interact with the virtual world.

We also plan to generalize our approach to make the whole system more general, for example to handle tall plant, hair, fur, feather, particle, character, rag-doll, and other types of animations. To generalize our approach while considering various gesture-based input devices, we plan to incorporate HCI (Human-Computer Interaction) principles and methodologies to improve the system.

The important HCI principles include: structure principle (organizing related things together and making unrelated things separated), simplicity principle (making simple and common tasks easy and providing shortcuts), visibility principle (making necessary options and materials clear without redundant information), feedback principle (providing users feedback of the actions, errors, or the status of the tasks), tolerance principle (tolerating users' mistakes and allowing undo/redo tasks), and reuse principle (reusing and remembering components and behaviors while maintaining consistency). We also plan to consider some HCI design approaches, for example, hierarchical problem solving, design by reuse (reusing existing designs for similar situations), design problem evolution (redefinition and relaxation of assumptions of the problems) and etc.

Our system design actually follows some HCI principles: structure principle (our method has a two-level structure), simplicity principle (the two-level approach makes the task in each level simpler), feedback principle (the authored global motions are replayed to give users feedback), the tolerance and reuse principles (users can edit and refine the authored results).

To generalize and improve our system, we plan to comprehensively study the HCI design principles and approaches and redesign the system. The possible improvements include:

adding shortcuts for creating some common animation effects by predefining them, providing more on-the-fly and helpful feedback to the users during the authoring, providing users with an interactive tutorial of the system, and conducting more comprehensive user studies (user needs study, system evaluation study, expert evaluation, and etc.).

Publications

1. **Kan Chen**, Henry Johan. Animating 3D Vegetation in Real-time Using a 2D Approach. In *I3D'15: Proceedings of the 2015 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 69-76, 2015. (Chapter 4)
2. **Kan Chen**, Xingzi Zhang, Shahzad Rasool, Henry Johan, Alexei Sourin. Multi-sensory Experience with Images. In *CW'14: Proceedings of the 2014 International Conference on Cyberworlds*, IEEE CPS, pages 55-62, 2014. (Chapter 3)
3. **Kan Chen**, Henry Johan. 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*, pages 244-251, 2013. (Chapter 3)
4. **Kan Chen**, Henry Johan, Wolfgang Mueller-Wittig. Simple and Efficient Example-based Texture Synthesis Using Tiling and Deformation. In *I3D'13: Proceedings of the 2013 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 145-152, 2013.
5. **Kan Chen**, Henry Johan. Interactive Authoring of Bending and Twisting Motions of Short Plants Using Hand Gestures. Accepted to *Computer Animation and Virtual Worlds*. (Chapter 5)

Appendix A

Bi-cubic Interpolation Coefficients

$$\vec{a}(0,0) = \vec{p}(1,1)$$

$$\vec{a}(0,1) = -0.5\vec{p}(1,0) + 0.5\vec{p}(1,2)$$

$$\vec{a}(0,2) = \vec{p}(1,0) - 2.5\vec{p}(1,1) + 2\vec{p}(1,2) - 0.5\vec{p}(1,3)$$

$$\vec{a}(0,3) = -0.5\vec{p}(1,0) + 1.5\vec{p}(1,1) - 1.5\vec{p}(1,2) + 0.5\vec{p}(1,3)$$

$$\vec{a}(1,0) = -0.5\vec{p}(0,1) + 0.5\vec{p}(2,1)$$

$$\vec{a}(1,1) = 0.25\vec{p}(0,0) - 0.25\vec{p}(0,2) - 0.25\vec{p}(2,0) + 0.25\vec{p}(2,2)$$

$$\begin{aligned}\vec{a}(1,2) &= -0.5\vec{p}(0,0) + 1.25\vec{p}(0,1) - \vec{p}(0,2) + 0.25\vec{p}(0,3) + 0.5\vec{p}(2,0) - 1.25\vec{p}(2,1) \\ &\quad + \vec{p}(2,2) - 0.25\vec{p}(2,3)\end{aligned}$$

$$\begin{aligned}\vec{a}(1,3) &= 0.25\vec{p}(0,0) - 0.75\vec{p}(0,1) + 0.75\vec{p}(0,2) - 0.25\vec{p}(0,3) - 0.25\vec{p}(2,0) + 0.75\vec{p}(2,1) \\ &\quad - 0.75\vec{p}(2,2) + 0.25\vec{p}(2,3)\end{aligned}$$

$$\vec{a}(2,0) = \vec{p}(0,1) - 2.5\vec{p}(1,1) + 2\vec{p}(2,1) - 0.5\vec{p}(3,1)$$

$$\begin{aligned}\vec{a}(2, 1) &= -0.5\vec{p}(0, 0) + 0.5\vec{p}(0, 2) + 1.25\vec{p}(1, 0) - 1.25\vec{p}(1, 2) - \vec{p}(2, 0) + \vec{p}(2, 2) \\ &+ 0.25\vec{p}(3, 0) - 0.25\vec{p}(3, 2)\end{aligned}$$

$$\begin{aligned}\vec{a}(2, 2) &= \vec{p}(0, 0) - 2.5\vec{p}(0, 1) + 2\vec{p}(0, 2) - 0.5\vec{p}(0, 3) - 2.5\vec{p}(1, 0) + 6.25\vec{p}(1, 1) \\ &- 5\vec{p}(1, 2) + 1.25\vec{p}(1, 3) + 2\vec{p}(2, 0) - 5\vec{p}(2, 1) + 4\vec{p}(2, 2) - \vec{p}(2, 3) \\ &- 0.5\vec{p}(3, 0) + 1.25\vec{p}(3, 1) - \vec{p}(3, 2) + 0.25\vec{p}(3, 3)\end{aligned}$$

$$\begin{aligned}\vec{a}(2, 3) &= -0.5\vec{p}(0, 0) + 1.5\vec{p}(0, 1) - 1.5\vec{p}(0, 2) + 0.5\vec{p}(0, 3) + 1.25\vec{p}(1, 0) - 3.75\vec{p}(1, 1) \\ &+ 3.75\vec{p}(1, 2) - 1.25\vec{p}(1, 3) - \vec{p}(2, 0) + 3\vec{p}(2, 1) - 3\vec{p}(2, 2) + \vec{p}(2, 3) \\ &+ 0.25\vec{p}(3, 0) - 0.75\vec{p}(3, 1) + 0.75\vec{p}(3, 2) - 0.25\vec{p}(3, 3)\end{aligned}$$

$$\vec{a}(3, 0) = -0.5\vec{p}(0, 1) + 1.5\vec{p}(1, 1) - 1.5\vec{p}(2, 1) + 0.5\vec{p}(3, 1)$$

$$\begin{aligned}\vec{a}(3, 1) &= 0.25\vec{p}(0, 0) - 0.25\vec{p}(0, 2) - 0.75\vec{p}(1, 0) + 0.75\vec{p}(1, 2) + 0.75\vec{p}(2, 0) - 0.75\vec{p}(2, 2) \\ &- 0.25\vec{p}(3, 0) + 0.25\vec{p}(3, 2)\end{aligned}$$

$$\begin{aligned}\vec{a}(3, 2) &= -0.5\vec{p}(0, 0) + 1.25\vec{p}(0, 1) - \vec{p}(0, 2) + 0.25\vec{p}(0, 3) + 1.5\vec{p}(1, 0) - 3.75\vec{p}(1, 1) \\ &+ 3\vec{p}(1, 2) - 0.75\vec{p}(1, 3) - 1.5\vec{p}(2, 0) + 3.75\vec{p}(2, 1) - 3\vec{p}(2, 2) + 0.75\vec{p}(2, 3) \\ &+ 0.5\vec{p}(3, 0) - 1.25\vec{p}(3, 1) + \vec{p}(3, 2) - 0.25\vec{p}(3, 3)\end{aligned}$$

$$\begin{aligned}\vec{a}(3, 3) &= 0.25\vec{p}(0, 0) - 0.75\vec{p}(0, 1) + 0.75\vec{p}(0, 2) - 0.25\vec{p}(0, 3) - 0.75\vec{p}(1, 0) + 2.25\vec{p}(1, 1) \\ &- 2.25\vec{p}(1, 2) + 0.75\vec{p}(1, 3) + 0.75\vec{p}(2, 0) - 2.25\vec{p}(2, 1) + 2.25\vec{p}(2, 2) - 0.75\vec{p}(2, 3) \\ &- 0.25\vec{p}(3, 0) + 0.75\vec{p}(3, 1) - 0.75\vec{p}(3, 2) + 0.25\vec{p}(3, 3)\end{aligned}$$

Appendix B

Procedural Skinning Weights

```
// The weights are computed based on the distances to the joints
// The distances are measured based on height
for  $i = 0, \dots, n - 1$  do
     $w_i \leftarrow 0$ 
    if  $v_y < 0$  then
         $w_0 \leftarrow 1$  // Under ground part
    end
    else
        // Find which bone it is related based on the height
        if  $J_{i_y} < v_y < J_{(i+1)_y}$  then
             $T \leftarrow \frac{v_y - J_{(i-2)_y}}{J_{i_y} - J_{(i-2)_y}}$ 
             $w_{i-1} \leftarrow \begin{cases} n/a, & \text{if } i == 0. \\ T, & \text{if } i == n - 1. \\ \frac{1.5}{T+0.5} - 1, & \text{otherwise.} \end{cases}$ 
             $w_i \leftarrow \begin{cases} 1 - T, & \text{if } i == 0. \\ 1 - T, & \text{if } i == n - 1. \\ 1 - \frac{0.5}{T+0.5}, & \text{otherwise.} \end{cases}$ 
             $w_{i+1} \leftarrow \begin{cases} T, & \text{if } i == 0. \\ n/a, & \text{if } i == n - 1. \\ 1 - \frac{1}{T+0.5}, & \text{otherwise.} \end{cases}$ 
        end
    end
end
```

Algorithm 1: Procedural weights

Appendix C

User Study Questionnaire

Dear user, the following questionnaire will not take longer than 5 minutes to be filled, and will help us to evaluate and, eventually, to improve the animation authoring interface you just experienced. The data provided will be analysed for academic purpose and will be treated as generic.

Section 1. Gender: ☐ M ☐ F

AGE: ☐ <18; ☐ 19 – 25; ☐ 26 – 35; ☐ 36 – 44; ☐ 45-54; ☐ >55

Field of expertise: _____

Years of Experience: ☐ < 3; ☐ 4 – 5; ☐ 6 – 8; ☐ 9 – 10; ☐ >11;

Are you familiar with 3D gestural interfaces (e.g. Kinect, Leap Motion)? ☐ Yes ☐ No

Are you familiar with 3D animation pipeline? ☐ Yes ☐ No

Are you familiar with 3D animation/modelling software (e.g. Maya)? ☐ Yes ☐ No

Section 2. For each of the statement below, please tick the box according to the following ranking: 5= I strongly Agree, 4= I agree, 3= I don't know, 2= I disagree, 1= I strongly disagree. Thanks for your precious time.

	5	4	3	2	1
1 Overall, I am satisfied with the animation authoring system.					

2	I can easily achieve the animation effects I desired.					
3.1	It is natural to describe a short plant's motions using hand.					
3.2	This animation authoring interface using hand to author the bending and twisting effects of short plants is intuitive.					
3.3	It is more intuitive to use hand to author the bending and twisting effects of short plants than using mouse and keyboard in existing 3D animation software (eg. Maya).					
4	It is helpful to watching the authored global animation while performing the authoring for the secondary animation.					
5	The two-levels authoring process is natural.					
6	The Leap Motion controller can well capture the motions of my hand and fingers.					
7	Overall, the animation results are visually plausible.					
8	The animation results generated by myself are visually plausible.					
9	The animation results generated by others are visually plausible.					
10	In the comparison with cartoons, the designed motions of short plants are similar as the cartoon animations.					
11	In the comparison with real plant motions, the generated animation results are similar with the real plant motions.					
12	In the comparison with SpeeTree result, the generated animation results are similar with the speeTree result.					

If you have any suggestion and/or observation, please feel free to share them with us here:

Question		Average Score	Standard Error
1	Overall, I am satisfied with the animation authoring system.	4.1	0.18
2	I can easily achieve the animation effects I desired.	4.2	0.2
3.1	It is natural to describe a short plant's motions using hand. (intuitiveness)	4.8	0.2
3.2	This animation authoring interface using hand to author the bending and twisting effects of short plants is intuitive. (intuitiveness)	4.6	0.16
3.3	It is more intuitive to use hand to author the bending and twisting effects of short plants than using mouse and keyboard in existing 3D animation software (e.g. Maya). (intuitiveness)	4.4	0.22
4	It is helpful to watching the authored global animation while performing the authoring for the secondary animation.	4.2	0.13
5	The two-levels authoring process is natural.	4.9	0.1
6	The Leap Motion controller can well capture the motions of my hand and fingers.	4.2	0.2
7	Overall, the animation results are visually plausible.	4.4	0.16
8	The animation results generated by myself are visually plausible. (user generated results)	4.1	0.12
9	The animation results generated by others are visually plausible. (user generated results)	4.1	0.34
10	In the comparison with cartoons, the designed motions of short plants are similar as the cartoon animations.	4.8	0.13
11	In the comparison with real plant motions, the generated animation results are similar with the real plant motions.	4.3	0.15
12	In the comparison with SpeedTree result, the generated animation results are similar with the SpeedTree result.	4.6	0.16

Bibliography

- [Ado] Adobe. Photoshop. *www.adobe.com*.
- [AK06] Yasuhiro Akagi and Katsuhiro Kitajima. Computer animation of swaying trees based on physical simulation. *Computers and Graphics*, 30(4):529–539, 2006.
- [Aut] Autodesk. 3DS Max. *www.autodesk.com*.
- [BCF⁺05] Stephan Behrendt, Carsten Colditz, Oliver Franzke, Johannes Kopf, and Oliver Deussen. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum*, 24(3):507–516, 2005.
- [Bea12] Andy Beane. *3D Animation Essentials*. SYBEX Inc., 2012.
- [BH02] Brook Bakay and Wolfgang Heidrich. Real-time animated grass. In *Proceedings of Eurographics (short paper) 2002*, 2002.
- [BHC⁺08] Florence Bertails, Sunil Hadap, Marie-Paule Cani, Ming Lin, Tae-Yong Kim, Steve Marschner, Kelly Ward, and Zoran Kačić-Alesić. Realistic hair simulation: animation and rendering. In *SIGGRAPH '08: ACM SIGGRAPH 2008 Courses*, pages 1–154, 2008.

- [BK04] J. Beaudoin and J. Keyser. Simulation levels of detail for plant motion. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004.
- [BMG06] Fabrice Boudon, Alexandre Meyer, and Christophe Godin. A survey on computer representations of trees for realistic and efficient rendering, 2006.
- [BN12] Eric Bruneton and Fabrice Neyret. Real-time realistic rendering and lighting of forests. *Computer Graphics Forum*, 31(2.1):373–382, 2012.
- [BPB06] Kévin Boulanger, Sumanta Pattanaik, and Kadi Bouatouch. Rendering grass terrains in real-time with dynamic lighting. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 46, 2006.
- [Bre] Paul Breeuwsma. Cubic Interpolation.
- [Bri08] Robert Bridson. *Fluid simulation for computer graphics*. A K Peters, 2008.
- [BW06] Sven Banisch and Charles A Wuthric. Making grass and fur move. *Journal of WSCG*, 14:25–32, 2006.
- [BZCJ09] Guanbo Bao, Xiaopeng Zhang, Wujun Che, and M. Jaeger. Billboards for tree simplification and real-time forest rendering. In *PMA '09: Proceedings of Third International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, pages 433–440, 2009.
- [CDN04] Florent Cohen, Philippe Decaudin, and Fabrice Neyret. Gpu-based lighting and shadowing of complex natural scenes. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 91, 2004.

- [CGZ⁺05] Yung-Yu Chuang, Dan B Goldman, Ke Colin Zheng, Brian Curless, David H. Salesin, and Richard Szeliski. Animating pictures with stochastic motion textures. *ACM Transactions on Graphics*, 24(3):853–860, 2005.
- [CJ10] Kan Chen and Henry Johan. Real-time continuum grass. In *VR '10: Proceedings of the 2010 IEEE Virtual Reality Conference*, pages 227–234, 2010.
- [CJMW13] Kan Chen, Henry Johan, and Wolfgang Mueller-Wittig. Simple and efficient example-based texture synthesis using tiling and deformation. In *I3D '13: Proceedings of 2013 Symposium on Interactive 3D Graphics and Games*, pages 145–152, 2013.
- [CKW⁺12] Guoning Chen, Vivek Kwatra, Li-Yi Wei, Charles D. Hansen, and Eugene Zhang. Design of 2D time-varying vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1717–1730, 2012.
- [Cyb] CyberGlove. CyberGlove Systems.
- [CZ⁺14] Kan Chen, Xingzi Zhang, , Shahzad Rasool, Henry Johan, and Alexei Sourin. Multisensory experience with images. In *CW '14: Proceedings of the 2014 International Conference on Cyberworlds*, pages 55–62, 2014.
- [Dac06] Carsten Dachsbacher. *Interactive terrain rendering: Towards realism with procedural models and graphics hardware*. PhD thesis, University of Stuttgart, 2006.

- [DDSD03] Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM Transactions on Graphics*, 22(3):689–696, 2003.
- [DFVHVR03] F. Di Fiore, W. Van Haevre, and F. Van Reeth. Rendering artistic and believable trees for cartoon animation. In *CGI '03: Proceedings of Computer Graphics International 2003*, pages 144–151, 2003.
- [DHBL⁺98] Oliver Deussen, Pat Hanrahan, Radomír Měch Bernd Lintermann, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 275–286, 1998.
- [DN04] Philippe Decaudin and Fabrice Neyret. Rendering forest scenes in real-time. In *Rendering Techniques '04: Proceedings of Eurographics Symposium on Rendering*, pages 93–102, 2004.
- [DN09] Philippe Decaudin and Fabrice Neyret. Volumetric Billboards. *Computer Graphics Forum*, 28(8):2079–2089, 2009.
- [Dog16] Naughty Dog. Uncharted 4, 2016.
- [DRBR09] Julien Diener, Mathieu Rodriguez, Lionel Baboud, and Lionel Reveret. Wind projection basis for real-time animation of trees. *Computer Graphics Forum*, 28(2):20–28, 2009.
- [dREF⁺88] Phillippe de Reffye, Claude Edelin, Jean Françon, Marc Jaeger, and Claude Puech. Plant models faithful to botanical structure and development. In

- SIGGRAPH '88: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, pages 151–158, 1988.
- [DRF06] Julien Diener, Lionel Reveret, and Eugene Fiume. Hierarchical retargeting of 2D motion fields to the animation of 3D plant models. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 187–195, 2006.
- [DYP03] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. *ACM Transactions on Graphics*, 22(3):409–416, 2003.
- [DZGL07] Qingqiong Deng, Xiaopeng Zhang, Sebastien Gay, and Xiangdong Lei. Continuous lod model of coniferous foliage. *IJVR*, 6(4):77–84, 2007.
- [EA13] EA. *Crysis 3*, 2013.
- [EL99] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV '99: Proceedings of the International Conference on Computer Vision*, pages 1033–1040, 1999.
- [EMF02] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics*, 21(3):736–744, 2002.
- [EMF03] Luis Carlos Yano Endo, Carlos Hitoshi Morimoto, and Antonio Elias Fabris. Real-time animation of underbrush. *Journal of WSCG (short paper)*, 11:41–48, 2003.
- [EMP⁺02] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and

- Steven Worley. *Texturing and modeling: a procedural approach*. Morgan Kaufmann, 2002.
- [ES03] George ElKoura and Karan Singh. Handrix: Animating the human hand. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 110–119, 2003.
- [FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer graphics and Interactive Techniques*, pages 23–30, 2001.
- [FLHS15] Zengzhi Fan, Hongwei Li, Karl Hillesland, and Bin Sheng. Simulation and rendering for millions of grass blades. In *I3D '15: Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, pages 55–60, 2015.
- [FM96] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. In *GI '96: Proceedings of the Conference on Graphics Interface 1996*, pages 204–212, 1996.
- [FM97] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 181–188, 1997.
- [FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 15–22, 2001.
- [FUM05] Anton L. Fuhrmann, Eike Umlauf, and Stephan Mantler. Extreme model

- simplification for forest rendering. In *NPH'05: Proceedings of the First Eurographics Conference on Natural Phenomena*, pages 57–67, 2005.
- [GCF01] Thomas Di Giacomo, Stéphane Capo, and François Faure. An interactive forest. In *Computer Animation and Simulation '01: Proceedings of the Eurographics Workshop*, pages 65–74, 2001.
- [GCCR11] Jesús Gumbau, Miguel Chover, Inmaculada Remolar, and Cristina Rebollo. View-dependent pruning for real-time rendering of trees. *Computers and Graphics*, 35(2):364–374, 2011.
- [Gen10] Weidong Geng. *The Algorithms and Principles of Non-photorealistic Graphics: Artistic Rendering and Cartoon Animation*. Springer-Verlag Berlin Heidelberg, 2010.
- [GGP⁺00] Michael Gleicher, F Sebastian Grassia, Zoran Popovic, S Rosenthal, and Jeffrey Thingvold. Motion editing: Principles, practice and promise. In *SIGGRAPH '00: ACM SIGGRAPH 2000 Courses*, 2000.
- [Gle98] Michael Gleicher. Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 33–42, 1998.
- [GMS⁺00] Wen Gao, Jiyong Ma, Shiguan Shan, Xilin Chen, Wei Zeng, Hongming Zhang, Jie Yan, and Jiangqin Wang. Handtalker: A multimodal dialog system using sign language and 3-d virtual human. In *Lecture Notes in Computer Science: Advances in Multimodal Interfaces — ICMI 2000*, volume 1948, pages 564–571. Springer Berlin Heidelberg, 2000.

- [GPR⁺03] Sylvain Guerraz, Frank Perbet, David Raulo, François Faure, and Marie-Paule Cani. A procedural approach to animate interactive natural sceneries. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents*, pages 73–78, 2003.
- [HAA97] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 225–232, 1997.
- [HCH12] Shaojun Hu, Norishige Chiba, and Dongjian He. Realistic animation of interactive trees. *The Visual Computer*, 28(6-8):859–868, 2012.
- [HFC09] Shaojun Hu, Tadahiro Fujimoto, and Norishige Chiba. 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, 2009.
- [HFR06] William Van Haevre, Fabian Di Fiore, and Frank Van Reeth. Physically-based driven tree animations. In *Eurographics Workshop on Natural Phenomena*, 2006.
- [HKK07] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Smoothed particle hydrodynamics on GPUs. In *CGI '07: Proceedings of Computer Graphics International 2007*, pages 63–70, 2007.
- [HKW09] Ralf Habel, Alexander Kusternig, and Michael Wimmer. Physically guided animation of trees. *Computer Graphics Forum*, 28(2):523–532, 2009.

- [HMT01] Sunil Hadap and Nadia Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338, 2001.
- [HRE⁺08] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics*, 27(3):27:1–27:11, 2008.
- [HWJ07] Ralf Habel, Michael Wimmer, and Stefan Jeschke. Instant animated grass. *Journal of WSCG*, 15:123–128, 2007.
- [IMH05] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, 2005.
- [JBPS11] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded bi-harmonic weights for real-time deformation. *ACM Transactions on Graphics*, 30(4):78:1–78:8, 2011.
- [JGGN15] Ming Jin, Dan Gopstein, Yotam Gingold, and Andrew Nealen. Animesh: Interleaved animation, modeling, and editing. *ACM Transactions on Graphics*, 34(6):207:1–207:8, 2015.
- [JSK09] Orthmann Jensn, Christof Rezk Salama, and Andreas Kolb. GPU-based responsive grass. *Journal of WSCG*, 17:65–72, 2009.
- [JTCW07] Doug L. James, Christopher D. Twigg, Andrew Cove, and Robert Y. Wang. Mesh ensemble motion graphs: Data-driven mesh animation with constraints. *ACM Transactions on Graphics*, 26(4):17:1–17:21, 2007.

- [KBKv09] Peter Krištof, Bedřich Beneš, Jaroslav Křivánek, and Ondřej Štava. Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum*, 28(2):219–228, 2009.
- [KCG⁺14] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. Draco: Bringing life to illustrations with kinetic textures. In *CHI '14: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 351–360, 2014.
- [KEBK05] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics*, 24(3):795–802, 2005.
- [KH84] James Thomas Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 165–174, 1984.
- [KK89] James Thomas Kajiya and Timothy Kay. Rendering fur with three dimensional textures. pages 271–280, 1989.
- [KP06] Paul G. Kry and Dinesh K. Pai. Interaction capture and synthesis. *ACM Transactions on Graphics*, 25(3):872–880, 2006.
- [KSE⁺03] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003.
- [Lay98] Kit Laybourne. *The Algorithms and Principles of Non-photorealistic*

- Graphics: Artistic Rendering and Cartoon Animation*. Three Rivers Press, 1998.
- [LD99] Bernd Lintermann and Oliver Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999.
- [LD14] Binh Huy Le and Zhigang Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics*, 33(4):84:1–84:10, 2014.
- [LDS⁺11] Chuan Li, Oliver Deussen, Yi-Zhe Song, Phil Willis, and Peter Hall. Modeling and generating moving trees from video. *ACM Transactions on Graphics*, 30(6):127:1–127:12, 2011.
- [Lea] Leap Motion, Inc. Leap Motion. www.leapmotion.com.
- [Lew87] J. P. Lewis. Generalized stochastic subdivision. *ACM Transactions on Graphics*, 6(3):167–190, 1987.
- [LPC⁺11] Yotam Livny, Soeren Pirk, Zhanglin Cheng, Feilong Yan, Oliver Deussen, Daniel Cohen-Or, and Baoquan Chen. Texture-lobes for tree modelling. *ACM Transactions on Graphics*, 30(4):53:1–53:10, 2011.
- [LPFH01] Jerome Lengyel, Emil Praun, Adam Finkelstein, and Hugues Hoppe. Real-time fur over arbitrary surfaces. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 227–232, 2001.
- [LPJ15] Jie Long, Bryce Porter, and Michael Jones. Animation of trees in wind

- using sparse motion capture data. *The Visual Computer*, 31(3):325–339, 2015.
- [LS12] Noah Lockwood and Karan Singh. Finger walking: Motion editing with contact-based hand performance. In *SCA '12: Proceedings of the 2012 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 43–52, 2012.
- [LWK03] Wei Li, Xiaoming Wei, and Arie Kaufman. Implementing lattice boltzmann computation on graphics hardware. *The Visual Computer*, 19(7-8):444–456, 2003.
- [LWW⁺07] Zhouchen Lin, Lifeng Wang, Yunbo Wang, S.B. Kang, and Tian Fang. High resolution animated scenes from stills. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):562–568, 2007.
- [LY12] Jing Liao and Jinhui Yu. Procedural models for cartoon cracks and fractures animations. *The Visual Computer*, 28(6-8):869–875, 2012.
- [LYP11] Jing Liao, Jinhui Yu, and John Patterson. Modeling ocean waves and interaction between objects and ocean water for cartoon animation. *Computer Animation and Virtual Worlds*, 22(2-3):81–89, 2011.
- [MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159, 2003.

- [MDN02] Ryo Miyazaki, Yoshinori Dobashi, and Tomoyuki Nishita. Simulation of cumuliform clouds based on computational fluid dynamics. In *Proceedings of Eurographics (short paper) 2002*, pages 405–410, 2002.
- [Mil86] Gavin S P Miller. The definition and rendering of terrain maps. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pages 39–48, 1986.
- [MN98] Alexandre Meyer and Fabrice Neyret. Interactive volumetric textures. In *Rendering Techniques '98: Proceedings of Eurographics Symposium on Rendering*, pages 157–168, 1998.
- [MWGZ09] Chongyang Ma, Li-Yi Wei, Baining Guo, and Kun Zhou. Motion field texture synthesis. *ACM Transactions on Graphics*, 28(5):110:1–110:8, 2009.
- [MWT11] Chongyang Ma, Li-Yi Wei, and Xin Tong. Discrete element textures. *ACM Transactions on Graphics*, 30(4):62:1–62:10, 2011.
- [NFD07] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics*, 26(3):88:1–88:8, 2007.
- [NFJ02] Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. *ACM Transactions on Graphics*, 21(3):721–728, 2002.
- [Nim] Nimble VR, Inc. Nimble VR. www.nimblevr.com.
- [NT04] Paul Noble and Wen Tang. Modelling and animating cartoon hair with

- nurbs surfaces. In *CGI '04: Proceedings of Computer Graphics International 2004*, pages 60–67, 2004.
- [OFT⁺03] Shin Ota, Tadahiro Fujimoto, Machiko Tamura, Kazunobu Muraoka, Kunihiko Fujita, and Norishige Chiba. $1/f$ noise-based real-time animation of trees swaying in wind fields. In *CGI '03: Proceedings of Computer Graphics International 2003*, pages 52–59, 2003.
- [OOI07] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 26:1–26:8, 2007.
- [OOSM14] Masaki Oshita, Hayato Oshima, Yuta Senju, and Syun Morishige. Character motion control by hands and principal component analysis. In *VRCAI '14: Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pages 171–179, 2014.
- [PC01] Frank Perbet and Marie-Paule Cani. Animating prairies in real-time. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 103–110, 2001.
- [Pel04] Kurt Pelzer. Rendering countless blades of waving grass. *GPU Gems*, pages 107–121, Randima Femando, 2004.
- [PJM94] Przemysław Prusinkiewicz, Mark James, and Radomír Měch. Synthetic topiary. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer graphics and Interactive Techniques*, pages 351–358, 1994.

- [PNH⁺14] Sören Pirk, Till Niese, Torsten Hädrich, Bedrich Benes, and Oliver Deussen. Windy trees: Computing stress response for developmental tree models. *ACM Transactions on Graphics*, 33(6):204:1–204:11, 2014.
- [PPH⁺96] Przemysław Prusinkiewicz, Przemysław Prusinkiewicz, Jim Hanan, Jim Hananz, Mark Hammel, Mark Hammely, and Radomír Mech. L-systems: from the theory to visual models of plants. *Machine Graphics and Vision*, 5(1/2):365–392, 1996.
- [PSK⁺12] Sören Pirk, Ondrej Stava, Julian Kratt, Michel Abdul Massih Said, Boris Neubert, Radomír Měch, Bedrich Benes, and Oliver Deussen. Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics*, 31(4):50:1–50:10, 2012.
- [Ram05] Rishi Ramraj. Dynamic grass simulation and other natural effects. *Game Programming Gems 5*, pages 411–419, Kim Pallister, 2005.
- [RB85] William T. Reeves and Ricki Blau. 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*, pages 313–322, 1985.
- [RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.
- [RMMD04] Alex Reche-Martinez, Ignacio Martin, and George Drettakis. Volumetric

- reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics*, 23(3):720–727, 2004.
- [RTK⁺14] Helge Rhodin, James Tompkin, Kwang In Kim, Varanasi Kiran, Hans-Peter Seidel, and Christian Theobalt. Interactive motion mapping for real-time character control. *Computer Graphics Forum*, 33(2):273–282, 2014.
- [SATK99] Mikio Shinya, Masakatsu Aoki, Ken Tsutsuguchi, and Naoya Kotani. Dynamic texture: physically based 2D animation. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, page 239, 1999.
- [SF92] Mikio Shinya and Alain Fournier. Stochastic motion—motion under the influence of wind. *Computer Graphics Forum*, 11(3):119–128, 1992.
- [SFK⁺08] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable MacCormack method. *Journal of Scientific Computing*, 35(2-3):350–371, 2008.
- [SHM06] Jung Shin, Michael Haller, and R. Mukundan. A stylized cartoon hair renderer. In *ACE '06: Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pages 64:1–64:9, 2006.
- [SKM⁺06] Eiji Sugisaki, Yosuke Kazama, Shigeo Morishima, Natsuko Tanaka, and Akiko Sato. Hair motion cloning from cartoon animation sequences. *Computer Animation and Virtual Worlds*, 17(3-4):491–499, 2006.

- [SLSG01] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics*, 20(2):67–94, 2001.
- [SMTH13] T. Shiratori, M. Mahler, W. Trezevant, and J.K. Hodgins. Expressing animated performances through puppeteering. In *3DUI '13: Proceedings of the IEEE Symposium on 3D User Interfaces*,, pages 59–66, 2013.
- [SO99] Tatsumi Sakaguchi and Jun Ohya. Modeling and animation of botanical trees for interactive virtual environments. In *VRST '99: Proceedings of the 1999 ACM Symposium on Virtual Reality Software and Technology*, pages 139–146, 1999.
- [SOL13] Yeongho Seol, Carol O’Sullivan, and Jehee Lee. Creature features: Online motion puppetry for non-human characters. In *SCA '13: Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 213–221, 2013.
- [Sou07] Tiago Sousa. Vegetation procedural animation and shading in Crysis. *GPU Gems 3*, pages 373–385, Hubert Nguyen, 2007.
- [SP09] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. *ACM Transactions on Graphics*, 28(3):40:1–40:6, 2009.
- [Spe] SpeedTree. *Interactive Data Visualization, Inc.*
- [SSSE00] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 489–498, 2000.

- [SSTM09] Eiji Sugisaki, Hock Soon Seah, Feng Tian, and Shigeo Morishima. Interactive shadowing for 2D anime. *Computer Animation and Virtual Worlds*, 20(2-3):395–404, 2009.
- [ST89] Richard Szeliski and Demetri Terzopoulos. From splines to fractals. In *SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, pages 51–60, 1989.
- [Sta97] Jos Stam. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum*, 16(3):159–164, 1997.
- [Sta99] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 121–128, 1999.
- [Stu98] David J. Sturman. Computer puppetry. *IEEE Computer Graphics and Applications*, 18(1):38–45, 1998.
- [Swo] Sword Heroes' Fate 3. Kingsoft Corporation.
- [SY05] Eiji Sugisaki and Yizhou Yu. Simulation-based cartoon hair animation. In *WSCG '05: Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 117–122, 2005.
- [TJ84] Frank Thomas and Ollie Johnston. Abbeville Press, 1984.
- [Tre] TressFX. *AMD*.

- [TZW⁺07] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Transactions on Graphics*, 26(3):87:1–87:8, 2007.
- [Vic] Vicon. Vicon, Inc.
- [Web08] J.P. Weber. Fast simulation of realistic trees. *IEEE Computer Graphics and Applications*, 28(3):67–75, 2008.
- [WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 479–488, 2000.
- [WLKT09] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117, 2009.
- [WP95] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 119–128, 1995.
- [WWS⁺15] Nkenge Wheatland, Yingying Wang, Huaguang Song, Michael Neff, Victor Zordan, and Sophie Jörg. State of the art in hand and finger modeling and animation. *Computer Graphics Forum*, 34(2):735–760, 2015.
- [WWZ⁺05] Changbo Wang, Zhangye Wang, Qi Zhou, Chengfang Song, Yu Guan, and Qunsheng Peng. Dynamic modeling and rendering of grass wagging in

- wind: Natural phenomena and special effects. *Computer Animation and Virtual Worlds*, 16(3-4):377–389, 2005.
- [Xfr] Xfrog. Xfrog, Inc. *www.xfrog.com*.
- [XGC07] Hui Xu, Nathan Gossett, and Baoquan Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics*, 26(4):article 19, 2007.
- [XWL⁺08] Xuemiao Xu, Liang Wan, Xiaopei Liu, Tien-Tsin Wong, Liansheng Wang, and Chi-Sing Leung. Animating animal motion from still. *ACM Transactions on Graphics*, 27(5):117:1–117:8, 2008.
- [ZB13] Yili Zhao and Jernej Barbič. Interactive authoring of simulation-ready plants. *ACM Transactions on Graphics*, 32(4):84:1–84:12, 2013.
- [ZCM11] Yubo Zhang, Carlos D. Correa, and Kwan-Liu Ma. Graph-based fire synthesis. In *SCA '11: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 187–194, 2011.
- [Zel04] Jeremy Zelnack. Vertex texture fetch water. *NVIDIA SDK White Paper*, 2004.
- [Zio07] Renaldas Zioma. GPU-generated procedural wind animation for trees. *GPU Gems 3*, pages 105–120, Hubert Nguyen, 2007.
- [ZST⁺06] Long Zhang, Chengfang Song, Qifeng Tan, Wei Chen, and Qunsheng Peng. Quasi-physical simulation of large-scale dynamic forest scenes. In *CGI*

'06: *Proceedings of Computer Graphics International 2006*, pages 735–742, 2006.

- [ZZJ⁺07] Long Zhang, Yubo Zhang, Zhongding Jiang, Luying Li, Wei Chen, and Qunsheng Peng. Precomputing data-driven tree animation. *Computer Animation and Virtual Worlds*, 18(4-5):371–382, 2007.