Making Grass and Fur Move

Sven Banisch and Charles A. Wüthrich
CoGVis/MMC – Faculty of Media
Bauhaus-University Weimar
D-99421 Weimar (GERMANY)
E-Mail: [sven.banisch|caw]@medien.uni-weimar.de

ABSTRACT

This paper introduces physical laws into the real–time animation of fur and grass. The main idea to achieve this, is to combine shell-based rendering with a mass-spring system. In a preprocessing step, a volume array is filled with the structure of fur and grass by a method based on exponential functions. The volumetric data is used to generate a series of two dimensional, semitransparent textures that encode the presence of hair or of the blades. In order to render the fur volume in real–time, these shell textures are applied to a series of layers extruded above the initial surface. Moving fur can be achieved by horizontally displacing these shell layers at runtime through a mass–spring mesh. Four different mass–spring topologies – different arrangements of masses and springs over the grass–covered surface – are introduced and used for animation. Two of them allow the shell layers to separate laterally, so that the "parting" of grass can be simulated. Performance observations prove mass-spring systems to be well-suited for the real–time simulation of fur and grass dynamics.

Keywords

Computer Graphics, Real–Time Animation, Physically–Based Simulation, Mass–Spring Systems

1 INTRODUCTION

Fur and grass are natural materials the real–time *rendering* of which has attracted many computer graphics researchers in the past years. High performance techniques for their fast rendering have been developed. A realistic animation, however, also requires an appropriate simulation of the dynamic behavior of the material. There are rare attempts focussing on this problem (e.g., [23, 14, 7]), but they do not yet use an adequate physical description, and therefore, do not yet provide a satisfactory solution. The motivation of this paper is to develop such a *physically–based* approach by introducing Newtons laws of motion into the real–time animation of fur and grass.

The most common method to render fur– and grass–covered surfaces in real–time is to represent the volumetric structure as a series of layers which are extruded above the initial surface and textured with semi–transparent hair textures. This method (called *shell–based rendering*) was introduced by Lengyel in 2000 [17] and was subsequently improved during the following years [16, 10, 7, 26]. A dynamical simulation of the surface structure can be achieved if the shell layers are horizontally displaced under the influence of external forces [16].

Journal of WSCG, ISSN 1213-6972, Vol.14, 2006
Plzen, Czech Republic.

The main issue of this paper is to analyze if, and in what way, *mass–spring systems* can be used to determine the shell layer displacement, hence, if such systems can form a suitable physical model for the real–time simulation of fur and grass dynamics.

A mass–spring system is a set of mass points which are linked by springs. They are usually arranged in a fixed topology. Mass–spring systems are mainly used for the real–time simulation of deformable objects, in particular, they are often involved into the simulation of cloth–like objects [3, 21, 8, 13, 29]. To research the applicability of mass–spring systems in the animation of fur and grass a topology of masses and springs (called *mass–spring topology*) is generated over the fur–covered surface represented by Lengyel's method. The shell method and the mass–spring system are combined to simulate movement.

The remainder of this paper is organized as follows: Section 2 presents a selection of previous related work. Section 3 describes basic algorithms involved into the dynamic grass simulation, and Section 4 then shows how these algorithms have been combined. The simulation results achieved by this combination are presented in Section 5. Finally, Section 6 sums up this work and points out issues for future development.

2 PREVIOUS WORK

Our approach combines two techniques which are well–developed and have been frequently used in the last years: the shell method as a real–time rendering technique for furry surfaces, and the mass–spring system used for the real–time simulation of deformations. In the following, very briefly, both techniques are separately considered.

Fur and Grass in Computer Graphics

For more than 20 years the image synthesis of grass has been an attractive topic of computer graphics. An early proposal by Reeves and Blau was based on particle systems. Particles were used to create the structure of grass as well as to render it [24]. Following the idea of representing hair as a set of particle traces, Kajiya and Kay introduced the texel – a three dimensional array that stores the hair data – and arranged these texels over an initial surface which is then rendered via ray tracing [12]. With both proposals high–quality image of grass and fur could be achieved at rendering times far beyond real–time.

In 1998, Meyer and Neyret [20] introduced interactive three dimensional textures using a method called slicing. This technique could be adapted to the more specific case of fur in 2000 by Lengyel [17]. His work proposed *shell-based rendering* for the animation of fur and grass in real–time. The shell method represents a furry surface as a series of concentrically ordered shell layers. The structure of fur and grass is encoded into the alpha channel of the shell textures, which are then applied one by one to their respective shell layers. Since 2000, there have been various authors (e.g., [16, 10, 7, 26]) addressing a more realistic real–time animation of fur and grass, basically, by improving the visual quality and the flexibility of shell-based rendering.

To the authors knowledge, there are only two attempts to a dynamic simulation of fur and grass displayed by a series of shell layers. The first one is a method developed for ATIs RADEON 8000 graphic card series which uses the programmable graphics hardware for the rendering as well as for the dynamical simulation [14]. The fur–covered object is rendered in two stages: one to compute the shell layer displacement, and a second one to draw the fur. The second idea, proposed in 2003 [7] and extended in 2004 [26], introduces so called "wind vectors" which, stored at each vertex of the object, determine the shell layer displacement. Global forces resulting from wind, gravity, motion and momentum are considered for the calculation the individual "wind vectors". Both attempts do not use the Newtonian laws of motion, and a "damping force" is added by "merging" the forces of two consecutive frames.

Mass–Spring Systems

Mass-spring systems are quite frequently used models to approximate the physical behavior of deformable objects (e.g., [22, 3, 8, 5, 15, 13, 21, 11]). They are easy to understand and to implement, highly parallelizable [29], and can achieve real–time rates. The nonrigid object is modeled as a set of mass points and springs in a fixed topology. The forces in between two masses are linearly approximated with Hooke's law and neglected for points that are not connected by a spring. These assumptions diminish the number of computations, so that mass-spring systems can simulate complex deformable objects at interactive rates.

Nevertheless, mass-spring systems do involve the solution of a system of ordinary differential equations (ODEs), because they base on Newtons fundamental law $\vec{F} = m\vec{a}$. In order to solve the equations of motion and to simulate such a system over time, a discretization in time has to be applied. A discrete time step is introduced and used for the numerical time integration of the equations "governing" the motion. Numerical time integration methods include explicit methods, such as the Runge Kutta method, and implicit predictor-corrector schemes.

The range of problems mass–spring systems have been used to solve is rather wide: from the simulation of cloth–like objects in virtual environments (e.g., [21, 8, 13, 29]), where detailed overviews are available [3, 18], over the simulation of rigid bodies attached to elastic ones [11], to the simulation of hair(e.g., [25, 1, 28, 27]), where the recent technological advances are summarized in [19]. Within this work, ways of adjusting masses and springs to shell–based rendering have been developed, so that the range of application of mass–spring systems is widened.

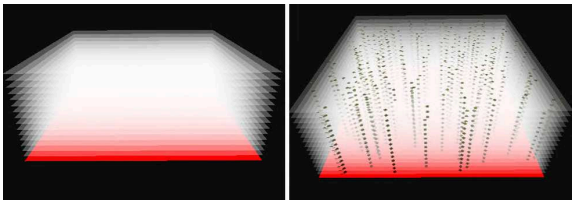
3 BASIC ALGORITHMS

This section briefly discusses the three basic components involved into the grass animation: shell–based rendering, shell texture generation and mass–spring systems.

Shell–Based Rendering

The implementation of the shell method was based on Lengyel's initial proposal [17] as well as on the enhancements proposed in [16]. A number of shell layers (ranging from 12 to 128 in this work) is extruded above the initial surface ("skin") of the object. This extrusion is determined by the surface normal vectors, specifying the direction, and the *inter–shell distance*, giving the distance between two consecutive layers.

Figure 1 illustrates the arrangement of the shell layers. To make the shell layers represent fur or grass, the corresponding semitransparent shell textures are applied to them. Figure 1b illustrates how this works.



(a) **(b)**

Figure 1: Shell layers are extruded above the initial surface **(a)**, and shell textures are applied to them **(b)**.

The shell textures only encode the structure of grass. Visual attributes, such as color and shadow, have to be added to obtain appealing renderings. In this work, per pixel lighting on programmable graphics hardware has been used to gain a higher flexibility in combining colors and evaluating different lighting models. The color is determined by the color of the initial surface, either by a "skin" texture or by per vertex colors. The shadow which individual blades throw onto each other (self–shadowing) is approximated by darkening the shell layers the closer they are to the initial surface[1]. Finally, a diffuse shading model, based on Lambert's cosine law, is applied to the furry surface. Figure 2 illustrates how adequate renderings of the furry dog are achieved by adding color, self–shadow and shading to the structure represented by concentric shell layers.

Shell Texture Generation

The generation of shell textures is of vital importance to the visual result of the shell method. The textures should encode the structure of fur and grass in a realistic fashion. The structure of fur and grass is rather simple: a large number of distinct individual elements is stochastically distributed over a surface. Most frequently, particle systems have been used to generate these individual elements (e.g., [24, 12, 17, 16, 10]).

There is, however, another generating method, which uses exponential functions to define the shape of the individual blade [4]. A large number of these curves is distributed within a volume data set which is then used to generate the shell textures by horizontally slicing it. To make the structure inhomogeneous, so to say more realistic, the stochastic parameters *length*, *direction of inclination* and *bending* are introduced and applied to each individual. This simple method proved to be well–suited for the generation of grass–like structures.

Following an idea thought by Kajiya and Kay in 1989 [12], we additionally generate an "undercoat" – a dense coat of short hair – in order to model an accurate structure of fur as well.

[1]The most distinctive characteristic of shadow within grass and fur is that it increases the further one approaches the ground. The method proposed in [2] is based on this characteristic.

Mass–Spring System

A mass-spring system is a fixed topology of mass points which are connected by springs. The animation of a system of n mass points requires the solution of the system of n second order ODEs

$$M\ddot{p} = F(t, p, \dot{p}). \tag{1}$$

The $n \times n$ dimensional matrix M stores the masses of all mass points on its diagonal, the n dimensional vector $p = (p_1, p_2, \ldots, p_i, \ldots, p_n)^T$ represents the positions of all points, and F is a function which describes the forces on the system at the time t. This system of n second order ODEs can be reduced to the system of $2n$ ODEs of first order

$$\frac{d}{dt}\begin{pmatrix} p \\ \dot{p} \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} p \\ v \end{pmatrix} = \begin{pmatrix} v \\ M^{-1}F(t, p, v) \end{pmatrix} \tag{2}$$

by substituting $v = \dot{p}$, where v is a n dimensional vector containing the velocities of all mass points.

There are several different methods to numerically solve such a system of ODEs. Explicit methods are very fast at the expense of accuracy and possible instabilities. Implicit methods are considered more stable, but are computationally more expensive. Since it was not clear how fast the mass–spring system has to be for the physical simulation of fur and grass, and how stable it behaves, both an explicit and an implicit *mass–spring solver* have been implemented. A detailed description of both solvers will not be presented within this paper. We refer the reader to standard literature on ODE solvers (e.g., [6, 3, 9]).

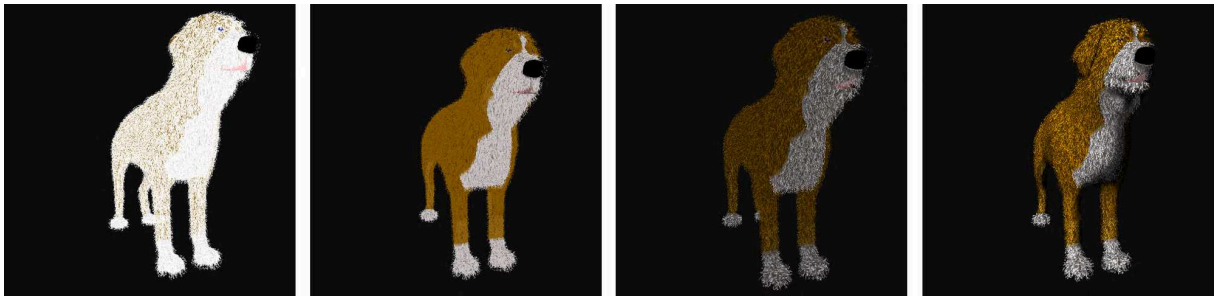
4 COMBINING THE METHODS

Grass represented by a series of shell layers moves if the shell layers are horizontally displaced [16]. All the previously proposed methods for moving grass animation use this characteristic [14, 7, 26]. The methods differ in the way the shell layer displacement is determined, hence in the physical laws modeling the grass movement.

In this work, mass–spring systems form this physical model. In order to combine such a system with Lengyel's rendering method, masses and springs are generated over the surface and attached to the shell layers. Different arrangements of masses and springs are called *mass–spring topologies*, and will be described first. After that, the method by which the movement of mass points is transformed into shell layer movement will be described.

Mass–Spring Topologies

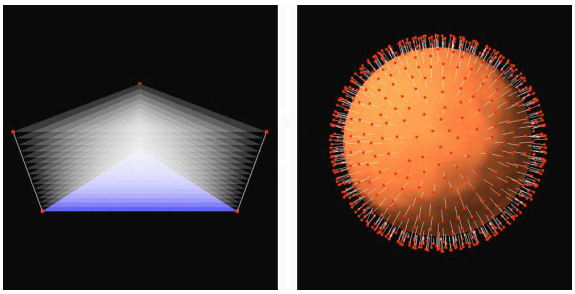
The first and most simple mass–spring topology is called *spring–stick topology*. At each vertex of the initial surface two mass points are generated – the first one attached to the initial surface by setting its mass



(a) **(b)** **(c)** **(d)**

Figure 2: Creating the image of fur by adding color **(b)**, self–shadowing **(c)** and a diffuse shading **(d)** to the fur structure **(a)**.

sufficiently large, and the second one connected to the uppermost shell layer. Both points are connected by a single spring. In Figure 3, the results of the spring–stick topology generated over an entire surface are illustrated.



(a) **(b)**

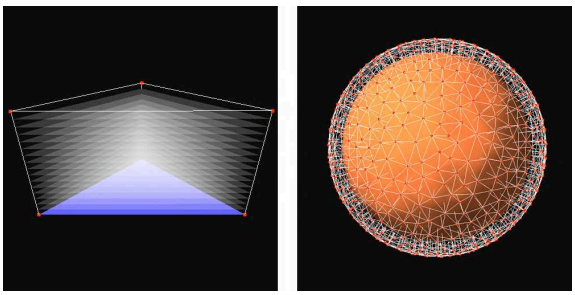
Figure 3: Spring–sticks applied to a single triangle **(a)** and to a sphere **(b)**.

The second topology is called *prism topology* since the masses and the springs form a prism on each surface triangle. Mass points are generated in the same way as it was done for the spring–stick topology. Also corresponding to the spring–sticks, the mass points are vertically linked by a spring. Additionally, we connect the mass points on the uppermost shell layer by generating springs along the edges of the uppermost layer. The resulting prisms are shown in Figure 4.

An interesting effect in grass animation is the simulation of *parting*. Parting means that fur and grass can be split, and that clusters of hair can move independently from other ones. This effect can be simulated if shell layers separate laterally[2]. The parting of fur and grass has been integrated into this work by implementing two additional topologies – *separable spring–sticks* and *separable prisms*.

Both separable mass–spring topologies are build by treating the triangles separately for the generation of mass points and springs. The triangles of the initial surface are handled one by one to attach a mass point

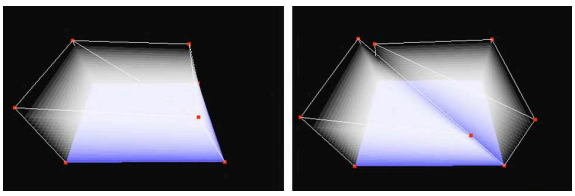
[2]In [16], Lengyel et al. already saw this possibility.



(a) **(b)**

Figure 4: Prisms applied to a single triangle **(a)** and to a sphere **(b)**.

to each of its vertices. Then, mass points are assigned to the vertices of the corresponding uppermost shell layer triangle. The way springs are generated depends on the respective mass-spring topology. In fact, we treat every triangle as being a distinct surface, as a result that the number of mass points and springs needed for an entire surface is increased using these topologies. The separable topologies, however, allow the simulation of parting since neighboring shell layer triangles are not connected. This is shown in Figure 5b.



(a) **(b)**

Figure 5: Neighboring shell layer triangles are connected **(a)**. The shell layers separate laterally **(b)**.

Of course, there are further possibilities of arranging masses and springs over a surface. More springs could be added to connect masses of the initial surface and masses of the uppermost shell layer. Also, mass points could be attached to a larger number of layers. This paper, however, only considers the

four mass–spring topologies spring–sticks, separable spring–sticks, prisms and separable prisms which have been introduced above.

The mass-spring topologies are generated in a preprocessing step. During the real time simulation, external forces are calculated and applied to the respective mass points. Then, the mass points positions are updated by the mass-spring solver. In the current state of the application, the external forces are with respect to gravity, wind and motion of the object. Additionally, it is possible to locally apply forces by user input.

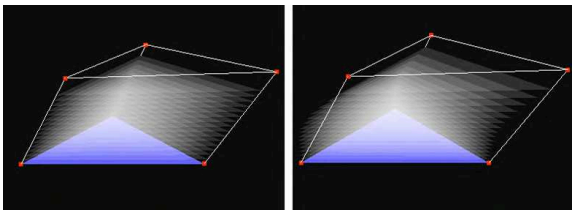
Shell Layer Displacement

Masses are attached to the initial surface and to the uppermost shell layer only. For this reason, a method to determine the shell layer displacement of layers which are not connected to a mass point is necessary. Such a method should approximate the natural bending of the blades. Bakay [7] proposes a technique which is based on trigonometric functions. However, this method is computationally expensive, and a new method has been developed.

During runtime, for every vertex, we first compute the *scale vector*

$$\vec{s}_{i,j} = \frac{p_j - p_i}{|p_j - p_i|}, \tag{3}$$

which is defined by the positions of its two mass points p_i, p_j. Since fur is usually rendered at large scales, it will often be sufficient to linearly displace the shell layers along this vector as shown in Figure 6a.



(a) **(b)**

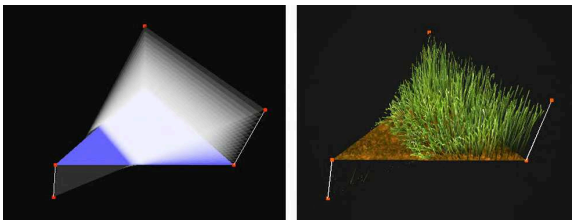
Figure 6: Linear displacement **(a)** and the approximation of the natural bending **(b)**.

In order to obtain an approximation matching better natural bending, the shell layer displacement is composed of a horizontal part $\vec{h}(l_i)$ and a vertical one along the vertex normal vector $\vec{n}$. The quantity of vertical displacement is determined by the fixed inter-shell distance. The horizontal displacement is defined by

$$\vec{h}(l_i) = \left(\frac{i}{n_{layers}}\right)^k (\vec{s} - \vec{n}), \tag{4}$$

which increases with the height of the layer l_i. Figure 6b shows how the shell layers separate from the straight line defined by the vertical springs. Note that the mass points, initially created on the uppermost layer, separate from it. This is because of the vertical displacement determined by the fixed value.

With combining the mass-spring system and the shell method as described, there is one general difficulty: once an external force has been applied to a mass point, it is free to move and does not return to its initial position. Moreover, it is possible that mass points move to the other side of the initial surface (inside the object). Figure 7 illustrates that fur flips to the wrong side of the surface if the mass points move accordingly.



(a) **(b)**

Figure 7: Grass flips to the wrong side of the surface.

In order to solve this problem, we use the normal vector $\vec{n}$ and the vector $\vec{s}$ to compute forces by which the mass points on the uppermost layer are forced to return to their initial position. The force $\vec{F}$ applied to the respective mass point is calculated by

$$\vec{F} = (\vec{n} - \vec{s})(1 - (\vec{n} \cdot \vec{s}))w, \tag{5}$$

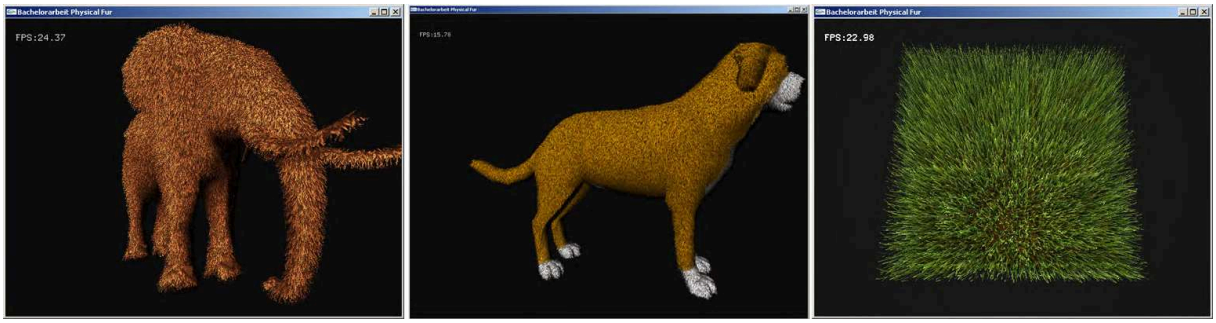
where the term $(\vec{n} - \vec{s})$ determines the direction of the force. The term $(1 - (\vec{n} \cdot \vec{s}))$ is zero in rest state, and increases with the angle formed by $\vec{n}$ and $\vec{s}$. Additionally, an adjustable weight w determines the magnitude of the force. With this procedure, the mass–spring topologies always return to their initial state, since the mass points on the uppermost layer are permanently forced to return to their initial position. Grass does therefore not flip to the wrong side of the surface.

5 RESULTS

Performance Analysis

The moving grass animation presented in this paper was developed and tested on a medium level platform (P4 with 1,7 GHz and 512 MB RAM, Nvidia GeForce 5700 FX with 256 MB). Benchmarks have been done for the three different models shown in Figure 8. Four topologies (spring-sticks, separable spring-sticks, prisms and separable prisms) are compared with each other regarding to the update times. The analysis also takes into account how much of the total time is needed to render the models. Moreover, the two types of mass–spring solver the explicit and the implicit variant are considered.

The first model is an elephant, consisting of 623 vertices and 1148 faces. It is rendered with 16 shell layers, a screen size of 640 $\times$ 480 and a screen coverage



(a) **(b)** **(c)**

Figure 8: An elephant **(a)**, a dog **(b)** and a square of grass **(c)** are used for the benchmarks.

(influencing the rendering speed since per pixel lighting is used) as shown in Figure 8a. The resolution of the shell textures, which is an interesting performance feature if many shell layers are used, is 128 × 128.

The size of the physical system depends on the mass–spring topology. 1247 mass points and 623 springs (one spring for each vertex) are generated if spring–sticks are used. In the largest case, using separable prisms, 6888 mass point and 6888 springs are generated and have to be updated during every frame.

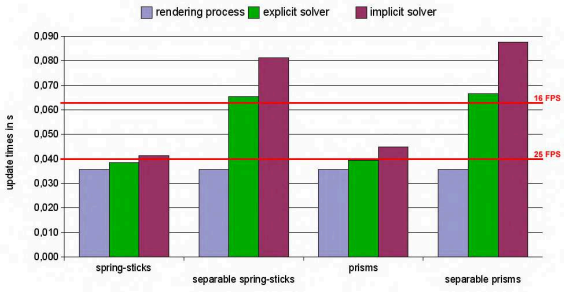


Figure 9: Update times of the elephant model.

The update times of the elephant model, shown in Figure 9, prove that the method is fast enough for real–time use. Spring–sticks and prisms work at update rates of 22 to 27 frames per second. Even when using the potentially slower implicit solution method, no significant slowdown can be noticed. Also, the separable topologies simulated using the explicit solver run at update rates which are still interactive. On this account, the proposed technique is faster than the method proposed in [26], even though the underlying physical laws are of higher complexity.

The dog model consists of 1872 vertices and 3220 faces, and is rendered using 12 shell layers. The size of the screen is 1024 × 768 and the resolution of the shell textures 128 × 128. The screen coverage is shown in Figure 8b. The benchmarking results of the second model are shown below.

The performance evaluation of the dog shows that the rendering process alone is near the limits of inter-

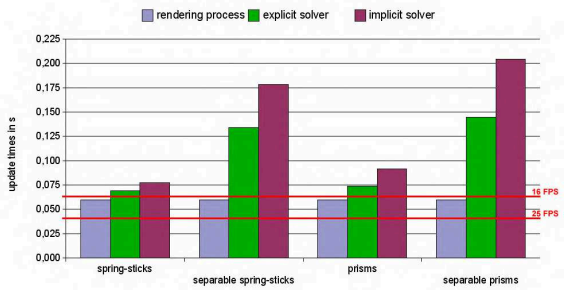


Figure 10: Update times of the dog model.

active application (≈ 17 FPS). Using the spring-sticks for the dynamical simulation does not noticeably slow down the system, and the interactive moving fur simulation of a model of more than 3000 faces and 1800 vertices is feasible, provided that the number of shell layers is reduced to 12.

The last model considered here is a grass model consisting of 13 vertices and 16 faces. It is rendered with 64 shell layers, a texture resolution of 64 × 64 and a screen size of 640 × 480. The amount the object covers of the screen is shown in Figure 8c. It is a crucial performance feature for this example since many shell layers are used to represent the grass.

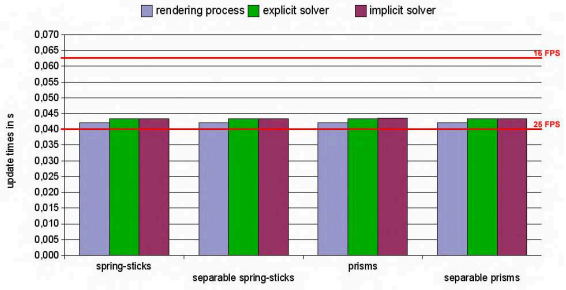


Figure 11: Update times of the grass model.

We can see only slight differences in the performance results. Nearly all the time needed to process the animation is required by the rendering, not by the physical system. Even the more complex mass-spring

topologies, simulated by the computationally more expensive implicit mass-spring solver, can be used for the physical simulation of long grass without causing the system to slow down.

The performance analysis shows that the real bottleneck of the system is the shell method, and not the physical simulation. This is obvious for the grass example, since the size of the mass-spring system is very small. But also for the moving fur simulation of the dog and the elephant, using non-separable topologies, the computational power needed for rendering the shell layers takes the greater part of the total performance.

Animation Quality

The usage of a mass-spring system in grass simulation allows a wide range of different effects. Such a system can react to arbitrarily applied forces, and realistic animation effects can be achieved if the forces that act on the system approximate the forces acting in the real world. Mass-spring systems, moreover, provide the possibility to control the dynamical behavior of fur by adapting the physical values which determine how the system behaves.

Figure 12: A close view on the parting of grass.

Unfortunately, it is rather difficult to present animation results in writing. Therefore, this paper is accompanied by a video, which shows furry objects in captured animation. Furthermore, in order to provide a rough idea of the quality of animation, Figure 13 shows a sequence of grass waving in the wind. Figure 12 presents a close view on the parting of grass, and gives an idea how this feature can enhance the realism of a grass animation.

6 CONCLUSIONS

A new method for the simulation of grass dynamics has been developed by combining the shell method and mass–spring systems, so that the range of application of mass–spring systems has been widened. Performance observations have proven the methods applicability of being used in a real–time context.

To conclude: mass–spring systems are well–suited to simulate dynamical effects of grass and fur.

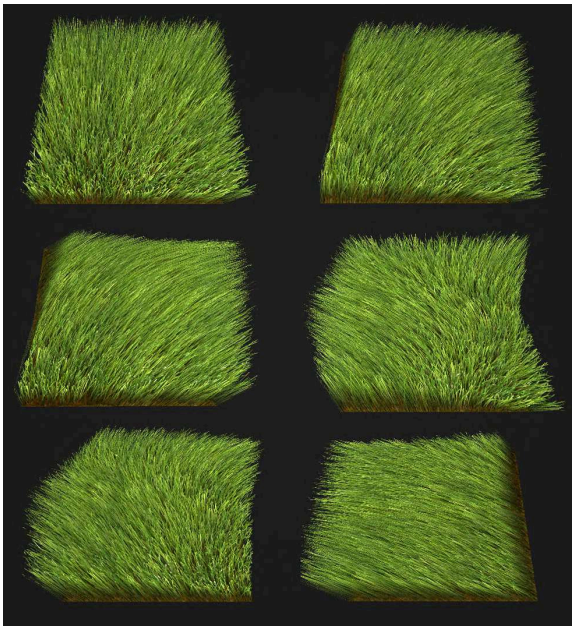


Figure 13: A sequence of grass waving in the wind.

Nevertheless there are several issues that could be addressed by future work.

First and foremost, it will be fruitful put some effort in optimizing the implementation, and ways to fully exploit all capabilities of new graphic cards hardware should be discussed. We believe that this can speed up the application greatly.

An important question to be answered concerns the dynamical effects resulting from motion of the object. There will be no need to calculate any external force, if motional changes of the object are directly transformed into motional changes of the mass points on the skin. Masses on the uppermost shell would react automatically. It is possible that there will be effects on stability of the mass-spring system. However, this way of handling motional changes of the object has to be implemented in future, since a lot of improvement to the moving fur animation is to be expected.

With the development of new graphics hardware optimized for ray tracing, it might also be possible in future, that fur and grass can be interactively rendered by three dimensional textures, and that the texel approach [12] will be taken back in consideration. Visual quality of fur and grass would be improved by great amounts. The proposed method can be adopted to simulate the physical behavior of structures represented by three dimensional textures.

Last but not least, it might be interesting to research more mass–spring topologies.

All in all, the presented technique yields good results and will be used for the simulation of fur and grass in future real–time productions, such as computer games or virtual puppetry.

7 ACKNOWLEDGMENTS

The authors wish to thank all participants of the project ”puppets & hands”. In particular, we thank Tobias Hofmann, Benjamin Schmidt, Uwe Hahne and Bernhard Bittdorf for their support.

References

[1] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proceedings of SIGGRAPH ’92*, pages 111–120, 1992.

[2] D. C. Banks. Illumination in diverse codimensions. In *SIGGRAPH ’94*, pages 327–334, 1994.

[3] D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH ’98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM Press, 1998.

[4] H. B. Bidasaria. A new method for modeling of hair-grass type textures. In *CSC ’95: Proceedings of the 1995 ACM 23rd annual conference on Computer science*, pages 109–113. ACM Press, 1995.

[5] D. Bourguignon and M.-P. Cani. Controlling anisotropy in mass-spring systems. In *Proceedings of the 11th Eurographics Workshop on Computer Animation and Simulation 2000*, Springer Computer Science, pages 113–123. Springer-Verlag, August 2000.

[6] I. Bronstein, K. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, Thun, third edition, 1997.

[7] M. Brook Maurice Bakay. Animating and lighting grass in real-time. Master’s thesis, The University Of British Columbia, 2003.

[8] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. In *Graphics Interface*, pages 1–8, June 1999.

[9] M. Hauth. *Visual Simulation of Deformable Models*. PhD thesis, Eberhard–Karls–Universität Tübingen, 2004.

[10] J. Isidoro and J. L. Mitchell. User customizable real–time fur, 2002. SIGGRAPH 2002 Technical Sketch, pp.273.

[11] J. Jansson and J. Vergeest. Combining deformable and rigid body mechanics simulation. *The Visual Computer*, pages 280–289, February 2003.

[12] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *SIGGRAPH ’89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 271–280. ACM Press, 1989.

[13] Y.-M. Kang and H.-G. Cho. Complex deformable objects in virtual reality. In *VRST ’02: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 49–56. ACM Press, 2002.

[14] T. Kano. Dynamic fur demo. ATI Developer: Source Code http://www.ati.com/developer/indexsc.html.

[15] U. G. Kühnapfel, H. K. Çakmak, and H. Maaß. Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers & Graphics*, 24(5):671–682, 2000.

[16] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *SI3D ’01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 227–232. ACM Press, 2001.

[17] J. E. Lengyel. Real-time hair. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 243–256. Springer-Verlag, 2000.

[18] N. Magnenat-Thalmann, F. Cordier, M. Keckeisen, S. Kimmerle, R. Klein, and J. Meseth. Simulation of Clothes for Real-time Applications. In *Proceedings of Eurographics 2004, Tutorial 1*, 2004.

[19] N. Magnenat-Thalmann, S. Hadap, and P. Kalra. State of art in hair simulation. *International Workshop on Human Modeling and Animation, Seoul, Korea*, pages 3–9, 2002.

[20] A. Meyer and F. Neyret. Interactive volumetric textures. In *Eurographics Rendering Workshop 1998*, pages 157–168. Eurographics, Springer Wein, July 1998.

[21] M. Meyer, G. Debunne, M. Desbrun, and A. H. Barr. Interactive animation of cloth-like objects in virtual reality. *Journal of Vizualisation and Computer Animation*, 2000.

[22] G. S. P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH ’88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 169–173. ACM Press, 1988.

[23] F. Perbet and M.-P. Cani. Animating prairies in real-time. In *ACM Interactive 3D Graphics*, USA, Mar 2001.

[24] W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH ’85 Comput. Graph.*, 19(3):313–322, 1985.

[25] R. E. Rosenblum, W. E. Carlson, and E. Tripp, III. Simulating the structure and dynamics of human hair: modelling, rendering and animation. 2(4):141–148, Oct.–Dec. 1991.

[26] G. Sheppard. Real–time rendering of fur. Bachelor thesis, The University of Sheffield, 2004.

[27] K. Ward and M. C. Lin. Adaptive grouping and subdivision for simulating hair dynamics. In *Pacific Conference on Computer Graphics and Applications*, pages 234–243, 2003.

[28] X. D. Yang, Z. Xu, J. Yang, and T. Wang. The cluster hair model. *Graphical Models*, 62(2):85–103, 2000.

[29] F. Zara, F. Faure, and J.-M. Vincent. Physical cloth simulation on a pc cluster. In *Parallel Graphics and Visualisation 2002*, 2002.