HAL
open science

A Procedural Approach to Animate Interactive Natural Sceneries

Guerraz Sylvain, Frank Perbet, David Raulo, François Faure, Marie-Paule Cani

To cite this version:

Guerraz Sylvain, Frank Perbet, David Raulo, François Faure, Marie-Paule Cani. A Procedural Approach to Animate Interactive Natural Sceneries. 16th International Conference on Computer Animation and Social Agents, CASA 2003, May 2003, New Brunswick, NJ, United States. inria-00516890

HAL Id: inria-00516890

https://inria.hal.science/inria-00516890v1

Submitted on 13 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Procedural Approach to Animate Interactive Natural Sceneries

Sylvain Guerraz, Frank Perbet, David Raulo, François Faure, Marie-Paule Cani
GRAVIR/IMAG, joint lab of CNRS, INPG, INRIA and UJF, www-gravir.imag.fr
Frank.Perbet@imag.fr, Francois.Faure@imag.fr, Marie-Paule.Cani@imag.fr

Abstract

This paper presents a method for animating and rendering an interactive natural scenery in real-time. It improves the prairie model of [11] by enabling the on-the-fly generation of blades of grass, at three different levels of detail, from user-specified density maps. A new animation function, the tread on grass, is defined to allow virtual objects or characters to crush the grass. The resulting grass model is incorporated into a more complex natural scene with the adjunction of trees that respond to the same wind. Results are illustrated by the real-time animation of autonomous virtual humans interacting with this natural scenery.

Keywords: animation, natural phenomena, procedural modelling, levels of detail, autonomous character

1. Introduction

Real-time applications such as driving simulators or video-games are more and more demanding complex outdoors scenes: an uneven terrain with animated grass, trees, rivers or water-falls, a sky with dancing clouds will be much more appealing than a conventional static environment. To feel really immerged, the user should not only be able to visit this animated scenery, but also to interact with it through the vehicle or the avatar he is controlling. This consists in both feeling the effect of the local slope and vegetation onto the avatar's motion and seeing the immediate consequences of this motion on the scenery, such as foot-prints or grass trodden down.

This paper shows that the use of a procedural approach for both modelling and animation can achieve this goal: it leads to the real-time animation of a large landscape, where grass and trees are generated on the fly and switched to the adequate level of detail according to the needs. Procedural animation is used to consistently model the action of wind on the vegetation, and the effect of moving objects crushing the grass. This work is illustrated by the interactive animation of a group of virtual humans wandering a natural play-ground.

2. Previous work

The first challenge in building interactive natural scenes is the number of elements (blades of grass, leaves, etc) to display. Since the camera may see each of these elements from very close to very far views, relying on multi-scale representations that provide different levels of detail is the best approach to render this complexity in real-time.

Most works on model generation at different levels of details (LODs) belong to a bottom-up approach: a fine mesh is progressively simplified into a coarse one [2]. Due to their sparse geometry, natural scenes are not well suited to this standard approach: trying to create and store the detailed geometry at the finest level before simplifying it would both be a waste of computational time and of memory, in addition to the fact that polygons are not adequate for approximating a natural scene at a coarse scale.

Thus, several alternative representations such as volume rendering [7, 8], specialized shaders [9, 10] and surface elements[13] have been adapted to render specific natural objects. These approaches take benefits of the detailed knowledge of the object to render for creating adapted levels of detail. The multi-scale representation is achieved in a smooth fashion by continuously changing the geometry and material properties of the displayed primitives while the camera is moving. However, pre-sampling and pre-filtering the objects makes most of these approaches difficult to use for animation purposes.

Real-time animation of natural scenes [11, 5, 1] has been recently achieved through the combination of some of these representations with procedural animation techniques. An adaptive mesh controlled by camera motion has been used for rendering real-time procedural wave-trains, filtered with the distance, resulting in an un-bounded interactive ocean [5]. Closer to our concerns, a volumetric representation proposed for fur [7, 8] has successfully been used to animate a field of grass moving under a procedural periodic wind field [1]: semi-transparent geometric layers, parallel

to the terrain, are displaced using vertex shaders according to pre-computed displacement vectors modelling the local wind direction. This approach is very convenient for animating a field of grass viewed from above for a flight simulator for instance. However, the need to pre-compute grass displacements and the use of a single level of detail prevent the use of this approach for the interactive animation of grass trodden down, viewed by walkers under the user control.

Our previous work [11] focussed on the rendering and animation of a prairie in real-time: grass samples were pre-computed at three level of details: 3D blades of grass, semi-transparent textures layers perpendicular to the ground's surface, and conventional 2D textures. Wind primitives interactively moving over the terrain generated procedural motion of the grass below them, whatever its current LOD. Smooth in-line transitions between LODs were generated according to camera motion. However, the need to pre-compute grass texture samples did not allow a sufficient control of the grass, which was just uniformly applied onto the terrain. The positioning of these samples caused artefacts when the camera was moving sideways. Moreover, restricting grass animation to the simulation of wind primitives did not allow much user interaction.

2.1. Overview

This paper is an extension of the later approach, and addresses most of its limitations: it first enables the on-line generation of grass primitives according to an arbitrary density map specified onto the terrain. The grass LODs are improved so that the rendering quality stays good wherever the camera moves. Trees animated under the same wind are added to increase the visual complexity of the scene. The animation is enhanced by enabling interaction between the vegetation and moving objects, which can be either autonomous or user-controlled.

Section 3 describes our improved grass model and the way we specify arbitrary grass density and add trees over the terrain.

Section 4 details the enhanced procedural animation method : in addition to gusts of wind applied to both grass and trees, a new animation primitive, the tread on grass, is proposed to model the interaction between the vegetation and moving objects.

Application to the real-time animation of a group of autonomous virtual humans interacting with vegetation on a sloppy terrain is presented in section 5.

Directions for future research are given in Section 6.

3. Procedural modelling of vegetation

3.1. Modelling fields of grass as a continuous medium

A reasonable field is characterized by a huge amount of blades of grass : approximately one billion by square kilometer. Creating and storing a geometric model for each of them would be impossible. We thus model the grass field as a continuous medium. To allow for some diversity, grass features such as height, density, colour and shape are allowed to vary over the field.

To achieve this, the grass field is tiled using a uniform grid with nodes every few meters. Bilinear interpolation is used to define grass feature values at any location from those stored at grid nodes. Modelling at the grid node level avoids the tedious work of creating each blade of grass and saves memory storage while still providing local control on then grass field. It thus provides the adequate level of abstraction.

The grass in each tile can be modelled and rendered at three different levels of detail depending on the current camera position. 3D blades of grass are generated in visible tiles close to the viewer; tiles which are far away are just covered by a 2D texture; the tiles in-between are rendered using a 2.5D representation similar the one used in [11].

The remainder of this section first explains how local grass density in each tile is used to compute the blades of grass positions. We then present our models for the three LODs and the way we generate transitions between them when the camera moves.

3.2. On the fly generation of grass

In [11], a square sample of grass made of specific 3D blades was pre-computed and stored. It was used to tile the closest part of the terrain and for pre-computing the grass textures to be used in the 2.5D representation, thus preventing any variation of the grass aspect or density.

In contrast, our new model generates grass on the fly from the used-specified grass density (represented as a grey scale map in figure 2, left). The grass is incrementally planted at random positions in each field tile according to the grass density at nodes. Each tile stores a seed for the random number generator. Reusing this seed at each frame ensures temporal coherence in the blade of grass locations. The number of blades of grass to generate is given by the average value of grass density at the four tile nodes. The random positions are biased according to the density gradient. This allows the blade density to vary smoothly (see figure 2, right).

3.3. The first LOD : 3D blades of grass

In this representation, used in tiles that are close from the viewpoint, each blade of grass is generated on the fly from the location of its base, the relative position of its extremity and two tangent vectors (the way to animate these parameters will be discussed in section 4). A Hermite curve, sampled and covered by several billboards, is used to do so.

Exactly keeping the same length for the blade during its bending motion under wind is difficult. A simple and efficient way to approximate constant length is to procedurally decrease the ordinate of the extremity of the blade when its absissa increases.

The blades are rendered using billboards. This is an easy way to decrease the geometrical complexity, since about 5 polygons are sufficient to render a convincing blade. The number of polygons per blade dynamically decreases when the distance from the viewer increases.

3.4. The 2.5D LOD : Semi-transparent vertical texture

Grass in tiles which are farther from the viewer is modelled using vertical polygons strips, covered with a semi-transparent texture representing blades of grass. We generate a single texture with several blades, each of them with a different alpha value. The alpha-test is then used to display as many blades as needed according to the density. Contrary with the 3D model, the density does not vary smoothly within a tile. The use of strips rather than a single vertical polygon will allow subsequent animation.

Contrary to our previous work [11], tiles are not covered by vertical strips of the same orientation. Four vertical strips are rather placed along the tile edges (see figure 1). This results into a much more view-independent geometry. This solution is more robust to camera motion over the scene, and will help achieving transitions between LODs. Note that similarly to [11], this representation is only adapted to grazing view angles, which is reasonable in the case of a walker's point of view.

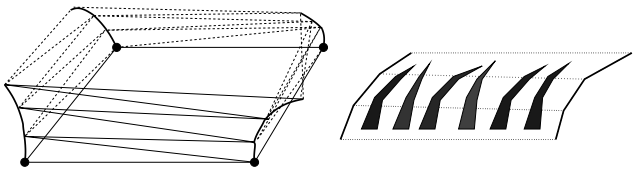

Figure 1. A terrain tile surrounded by semi-transparent textures.

When the camera moves, tiles of grass have to be able to switch back and forth between 3D to 2.5D. To achieve these transitions without visual popping, the geometry is continuously morphed and blended. We found out that, for grass, insuring that the geometry is exactly preserved, as was done in [11], is unnecessary. Indeed, the geometry of a field of grass is very complex and the eye does not detect easily such precise artefacts. Moreover, using simplified transitions greatly decreases the complexity of the structures used in this algorithm. The transitions between 3D and 2.5D are achieved using the following method: the 3D blades inside a tile are progressively displaced toward the edge of the tile, by changing the position of their base; meanwhile, the semi-transparent textured polygons are slowly growing up. The reverse is performed for going from 2.5 representation to 3D.

3.5. the last LOD : flat texture

A grass-like texture is applied onto the ground to fill the empty spaces between the blades of grass in the 3D and 2.5D representations. This texture is also used to represent the grass at the roughest level. The transition from and to the 2.5D level is made as in [11] by progressively lifting the semi-transparent polygons from the ground.

3.6. Trees

Our natural scenery is completed by the addition of trees. Their geometry is built on-the-fly similarly to the grass, using the method in [3]. The trees are procedurally generated at each frame using a small amount of data such as seeds for random number generators. Contrary to blades of grass, each individual tree stores its own data. This limits the number of trees but gives more control to the artist.

4. Animation

4.1. Procedural animation

To animate the vegetation, we use animation primitives associated with 2D masks, as in [11]. These primitives move over the terrain and send information to each node covered by their mask.The other nodes are not animated. Each primitive influences one or several features of a node, for instance, amplitude and direction of bending.

At each frame, the information received by a node from different animation primitives is either averaged (e.g. colour) or summed (e.g. displacement), depending on its nature. This simple ways to combine information could probably be improved, but already yield aesthetic animation at low cost: Since grass is generated on the fly from the current features at nodes, is adequately moves under the primitives actions, whatever its current LOD (in the 2.5D case,

displacements are applied to the extremities of each polygon strip). Similarly, animation primitives send information to trees, which can be animated using the different methods in [3].

The available animation primitives include several wind functions which were already introduced in [11]: a gust of wind, a tornado and a soft breeze. Their combination results in quite convincing animations of an outdoors scene under wind (fig.3).

4.2. Treading the grass

To offer better immersion in a video-game context, the landscape model should react to the motions of objects, vehicles and characters, possibly interactively controlled by the user. We illustrate this kind of effect by providing a new animation primitive, "treading", which only acts on the grass.

Treading is defined as a specific animation primitive attached to each moving object (for instance, the feet of a walking character). This animation primitive makes the grass locally bend and crush. It also controls the time that blades of grass take to slowly recover their initial shape.

At each time step, a line segment from the previous to the current position of the moving object is used to generate a *scalar field* with a finite radius of influence around it, similarly to what is done in the skeleton-based implicit surfaces approach.

To define a smooth field without bulges from this polyline, we use the closed-form convolution field proposed in [6]. The field applied by a trajectory the segment $[P_0, P_1]$ at a given node P is given by: $F(P) = \frac{\sin(\alpha 1) - \sin(\alpha 2)}{distance^2(P,H)}$ where H is the projection of P on $[P_0, P_1]$, and α_1 (respectively α_2) is the angle $([P, H], [P, P_0])$ (respectively $([P, H], [P, P_1])$).

The field at the neighbouring nodes is recursively computed until it goes under a given threshold. This field value is combined with a treading direction to compute the local grass displacement vector: Let C be a unit vector along a given trajectory segment, directed towards the direction of motion; N be the normal vector, perpendicular to this trajectory segment; and A be the vector from the first extremity of the segment to the node P. The treading vector expressing grass displacement at P is then computed as:

$E(P) = F(P)(sign(A.N).N + C)$

This vector pushes the grass either to the right or to the left, and along the direction of motion. The field amplitude is tuned such that a maximally crushed blade is almost horizontal. The nodes can undergo several crunching actions at different times. When a node is already crunched, only the vertical part of the crushing vector is updated. This prevents crushed blades from artificially rotating.

Each time a node undergoes crunching, it stores the current time and the updated crunching vectors. These values are used to (re)start an interpolation toward zero over a given time interval, allowing the blades to progressively recover their uncrunched shape.

The action of a user-controlled rolling-stone crushing the grass is depicted in figure 4.

5. Application : autonomous characters interacting with the vegetation

We have populated the landscape with a group of autonomous virtual humans. Their behaviour basically consists in walking across the terrain and avoiding the obstacles while treading on the grass. In addition to the other characters and the trees, the obstacles include "virtual walls", i.e. artificial landscape limits that help keeping the characters in the virtual area.

Each character is controlled through a behaviour model composed of three basic components : (1) a system for simulating perception, (2) a decisional process for navigation, (3) a real-time motion controller.

5.1. Simulated perception

The simulated perception is designed as a filtering process, gathering information in the environment and summing it up as *sensor variables*. For the purpose of reactive obstacle avoidance, the only obstacles of interest are those in the immediate character's surrounding. This can be further simplified by taking into account only the nearest obstacle in a short-range neighbourhood. Thus, the meaningful information can be abstracted as the distance d between this obstacle and the virtual human, and its relative direction $theta$. In order to cheaply compute these values (d, θ), a simplified scene-graph is maintained, containing only the bounding boxes of the trees' trunks and those of the remaining characters, plus the virtual walls that materialize the limits of the scene. Then an improved version of the algorithm of Gilbert-Johnson-Keerthi (GJK) is used to compute the shortest distances between the characters and these possible obstacles ([4, 14]), and we finally keep the nearest one intersecting the character's sensor-range (see figure 5).

5.2. Reactive avoidance behaviour

The rationale of the decision process is to transform the sensor information (d, θ) into the *motor command* (v, ϕ), where v is the desired linear velocity and ϕ the desired leading direction of the character. The value of v is computed so as to slow down when approaching an obstacle, or is set to a canonical walking speed when there is no obstacle. ϕ is

computed such that the character turns in the opposite direction than θ. Finally, a gaussian noise is added to these values in order to mimic a more natural behaviour, with a standard deviation inversely proportional to d so that the behaviour is better constrained in presence of a very close obstacle. Actually, this simplified behavior derives from a more general framework using probabilistic reasoning for animating Virtual Humans, described in [12].

5.3. Motion controller

The purpose of the real-time motion controller is twofold. First, it smoothly adapts the position and orientation of the character according to the motor command (given by the decision process) and the terrain's local slope while mimicking natural dynamics; then it derives the current character's pose. The locomotion gait is based on key-frames obtained from cyclic motion data of a human character walking. These key-frames are real-time interpolated using Bezier-curves in order to provide the skeleton state. In this process, the character current velocity is used to adapt the motion amplitude of the skeleton.

5.4. Action on the environment

Finally, the motion of the virtual humans has a direct consequence on the Vegetation animation, materialized by grass trodden down. This effect is achieved by attaching a treading primitive to each character. The character trajectories then feed the procedural animation process described in section 4.2. Separated feet trajectories need a higher terrain node resolution.

6. Conclusion and future work

This paper has presented a very flexible method for generating animated outdoors scenes in real-time. Animated vegetation elements are generated on the fly, at the adequate levels of detail, thus enabling camera motion over a large scenery. Procedural animation has successfully been used to control wind primitives blowing over grass and trees, and to simulate the interaction between this vegetation and moving objects. The animation of autonomous virtual humans treading grass down and avoiding obstacles while walking over a sloppy terrain has illustrated the interest of this approach for real-time applications such as simulators or video-games. We believe that the high level control of animation provided here is well suited for video games, and is a good compromise for easily animating a large scene without having to specify small scale motion.

In future work, we plan to use a multiresolution terrain in order to improve the precision of the treads on the grass. We also think about including several grass or flower species. This is easy in procedureal 3D but more difficult with our other models.

As in [11], our grass model is better suited to low views, close to the ground, due to the use of vertical textured polygons in the 2.5D representation: These vertical polygons will produce visual artefact each time a distant sloppy hill is displayed, since grass it then viewed from above. As a future work, we are planning to modify the third level of detail in the grass representation in order to generate semi-transparent texture layers parallel to the ground instead of the 2D texture we are currently using. We would then switch to this representation each time grass is viewed from above, the number of layers being adapted to the viewing distance. This method would avoid visual artefacts and allow grass animation through displacement textures, similarly to the method in [1].

Acknowledgments: The authors would like to thank Thomas Di Giacomo and Stphane Capo for enabling us to re-use their work on the interactive animation of trees. Thanks to Infograme and to the CNC for sponsoring this project through the PRIAM program.

References

[1] B. Bakay, P. Lalonde, and W. Heidrich. Real-time animated grass. In *Eurographics 2002*. I. Navazo and P.Slusallek, 2002.

[2] M. Garland. Multiresolution modeling: Survey and future opportunities, 1999.

[3] T. D. Giacomo, S. Capo, and F. Faure. An interactive forest. In M.-P. Cani, N. Magnenat-Thalmann, and D. Thalmann, editors, *Eurographics Workshop on Computer Animation and Simulation*, pages 65–74. Springer, sept. 2001. Manchester.

[4] E. Gilbert, D. Johnson, and S. Keerthi. A fast procedure for computing the distance between complex objects in three dimensional space. *IEEE Journal of Robotics and Automation*, 4, 1988.

[5] D. Hinsinger, F. Neyret, and M.-P. Cani. Interactive animation of ocean waves. In *Symposium on Computer Animation*, july 2002.

[6] S. Hornus, A. Angelidis, and M.-P. Cani. Implicit modelling using subdivision-curves. *The Visual Computer*, 2002.

[7] J. Lengyel. Real-time hair. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 243–256. Eurographics, June 2000. ISBN 3-211-83535-0.

[8] J. E. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *2001 ACM Symposium on Interactive 3D Graphics*, pages 227–232, March 2001. ISBN 1-58113-292-1.

[9] A. Meyer and F. Neyret. Multiscale shaders for the efficient realistic rendering of pine-trees. In *Graphics Interface*, pages 137–144. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 2000. ISBN.

[10] A. Meyer, F. Neyret, and P. Poulin. Interactive rendering of trees with shading and shadows. In *Eurographics Workshop on Rendering*, Jul 2001.

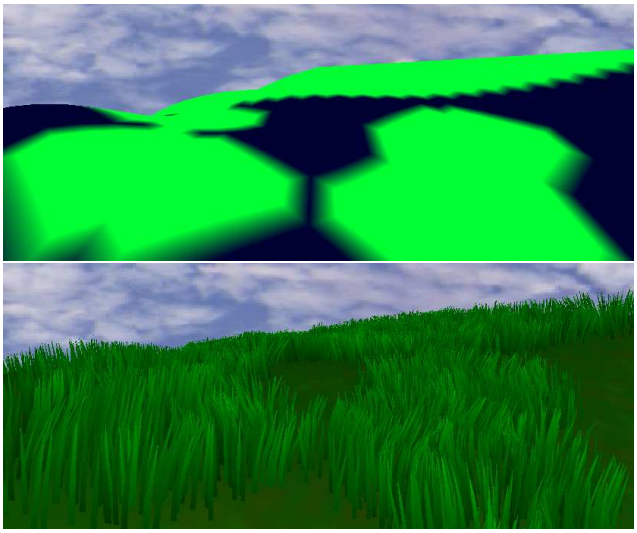

Figure 2. Grass density. On top, the grass density map stored as density values at nodes. On bottom, the resulting on-the-fly generated grass.

Figure 3. An animated scenery.

[11] F. Perbet and M.-P. Cani. Animating prairies in real-time. In S. N. Spencer, editor, *Proceedings of the Conference on the 2001 Symposium on interactive 3D Graphics*. Eurographics, ACM Press, 2001.

[12] D. Raulo. *Autonomie du mouvement pour des agents en ralit virtuelle*. PhD thesis, Institut National Polytechnique de Grenoble, FRANCE, 2003.

[13] M. Stamminger and G. Drettakis. Interactive sampling and rendering for complex and procedural geometry. In K. Myskowski and S. Gortler, editors, *Rendering Techniques 2001 (Proceedings of the Eurographics Workshop on Rendering 01)*, 12th Eurographics workshop on Rendering. Eurographics, Springer Verlag, 2001.

[14] K. Sundaraj, D. d'Aulignac, and E. Mazer. A new algorithm for computing minimum distance. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Takamatsu (JP), November 2000.

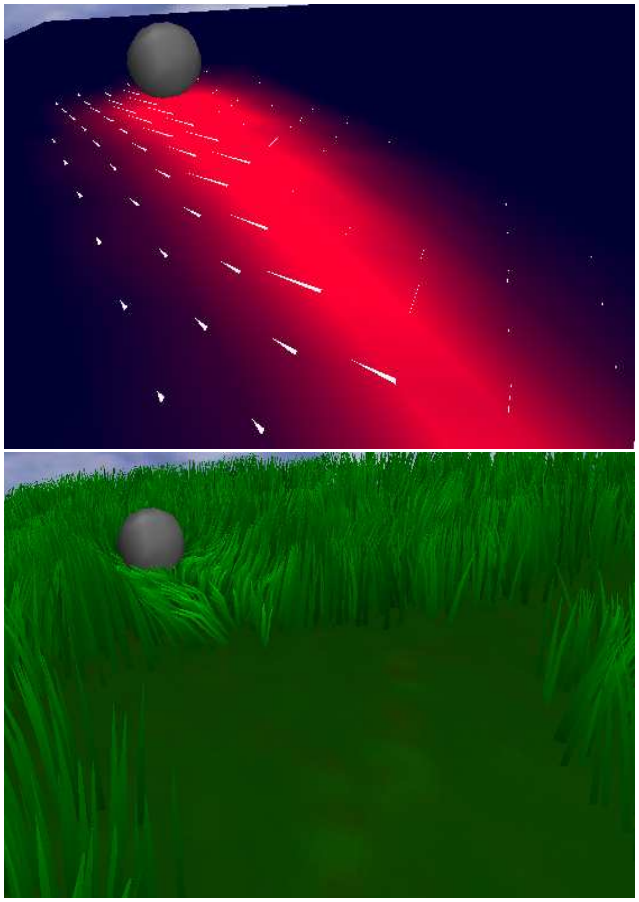

Figure 4. Treading on grass. On top, the field values and crushing vectors generated by the trajectory of a rolling stone. On bottom, the resulting grass animation.

Figure 5. Simulated perception of the characters. The red circle represents the sensor-range, and the red line segment show the nearest obstacle.