Real Time Tree Sketching

Celso Campos[1], Ricardo Quirós[2], Joaquin Huerta[2], Emilio Camahort[3], Roberto Vivó[3], and Javier Lluch[3]

[1] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Vigo, Spain
ccampos@ei.uvigo.es
[2] Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I, Spain
{quiros, huerta }@lsi.uji.es
[3] Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain
{camahort, rvivo, jlluch}@dsic.upv.es

Abstract. Modeling and rendering of synthetic plants and trees has always received a lot of attention from computer graphics practitioners. Recent advances in plant and tree modeling have made it possible to generate and render very complex scenes. Models developed so far allow low quality and photorealistic rendering as well as a fine control on the amount of geometry sent to the graphics pipeline. Recently, non-photorealistic rendering techniques have been proposed as an alternative to traditional rendering. In this paper we present a method for interactive rendering of vegetation silhouettes. Our goal is to expressively render plant and tree models. Previous methods are either too slow for real-time rendering or they do not maintain the general appearance of a given vegetable species. We solve these two problems in our work.

1 Introduction

Modeling and rendering of synthetic plants and trees has always received a lot of attention from computer graphics practitioners. Plant and tree models are fundamental to the representation of forests, gardens and interior scenes. Still, these models are geometrically complex and their description and rendering require a lot of resources in traditional computer graphics. A tree model may contain hundreds of thousands of polygons, and a forest scene may contain thousands of trees. Therefore, interactive rendering of such scenes requires specific modeling and acceleration techniques for applications such as outdoor walkthroughs and fly-by´s.

Recent advances in plant and tree modeling have made it possible to generate and render very complex scenes. The models developed so far allow low quality and photorealistic quality rendering as well as a fine control on the amount of geometry sent to the graphics pipeline. The most important plant and tree models are based on L-systems [1] [2].

Recently, non-photorealistic rendering techniques have been proposed as an alternative to traditional rendering [3]. Their goal is to increase the expressiveness of rendering, using techniques similar to those used in the arts [4], in animated movies (toon

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 197–204, 2004.

shading), and in sketches representing buildings and interiors [5]. All these applications usually include plant and tree models. Hence, there is a need for specific techniques to efficiently render expressive images from these models.

In this paper we present a method for interactive rendering of vegetation silhouettes. Our goal is to expressively render plant and tree models. Previous methods are either too slow for real-time rendering or they do not maintain the general appearance of a given vegetable species. We solve these two problems in our work. First, we survey previous work in non-photorealistic rendering of plants and trees. Then, we present a method that generates simplified tree models and we describe our silhouette rendering algorithm. We conclude our paper with some results and directions for future work.

2 Previous Work

2.1 Non-photorealistic Rendering of Plants and Trees

The structural complexity of plants and trees requires using specific techniques to render them non-photorealistically. The first methods for automatic illustration of vegetable species were introduced by Yessios [6] and Sasada [7]. They both produce tree sketches for architectural applications.

Kowalski et al. [8] also create abstract tree sketches using geometric primitives that approximate the tree's foliage. Those primitives are used in a two-step rendering algorithm. The first step creates gray-scale reference images of the trees. The second step improves the references by adding so-called *graftals*, small objects that represent leaves, grass or hair.

Markosian et al. [9] improve on Kowalki's work by using a static scheme for graftal definition during the modeling stage. At rendering time a graftal may or may not be rendered depending on the viewing parameters. Some graftals, known as tufts, are stored using a multiresolution representation that allows any graftal to become a set of tufts when the viewer is close enough. Other improvements take advantage of frame-to-frame coherence and add control to the appearance and behavior of the graftals. Still, the rendering algorithm is slow for very complex scenes.

Deussen [10] presents a method that creates pen-and-ink illustrations of plants and trees. The method starts with a detailed tree model that includes a branch and leaf skeleton. This skeleton is later used to compute the silhouettes necessary to draw the tree's contour. Rendering is accomplished by combining a large set of primitives instead of using graftals. This allows the representation of specific plants and trees and not just generic trees like those in [8] and [9].

More recently, Di Fiore [11] proposes a method that renders cartoon shaded trees. It uses tree models generated from L-systems. The models contain no leaves, but only the hierarchical structure of the trunk and branches. Given that information, the artist develops a picture library to represent branches and leaf groups. The final image is obtained by rendering the pictures corresponding to the branches and adding at the branch joints the pictures that represent the leaf groups.

2.2 Stroke-Based Rendering of Plants and Trees

[12] and [13] present a stroke-based method for non-photorealistic rendering of plants and trees. This method improves on Meier's algorithm [4] by supporting interactive frame rates. The method can be applied to videogames, virtual walkthroughs and other real-time computer graphics applications.

The method models trees using random parametric L-systems (RL-systems) [14]. These are an improvement on L-systems that associates to each system a set of random variables. This approach has several advantages over the surface patch algorithm of Meier. For example, it supports the simultaneous generation of both the tree's geometry and its stroke particles for future rendering.

The stroke particles are distributed using the same RL-system that is used for modeling the tree's geometry. To achieve this goal we use a *shape instantiation* process. This process represents every instantiable object, a branch or leaf, using both 3D geometry and a cloud of strokes. The latter are used for expressive rendering. Fig. 1 shows some results obtained using our software to render different plants and trees.

Fig. 1. Stroke-based rendering results

3 Tree Generalization

We present our approach to generalizing a geometric tree model to an abstract model that maintains the visual appearance of the tree and supports its silhouette-based rendering. In the following Section we describe how we render these models.

Modeling using RL-systems requires two steps: *system derivation* and *graphical interpretation*. Given a system, derivation generates a *parametric chain*. This chain is interpreted to obtain the geometric model to be rendered. In this paper, we use an RL-system that generates a ternary tree model [12].

RL-systems allow us to model plants and trees keeping their structural complexity. This is good for photorealistic rendering, where a lot of detail may be required. But this may be too much detail for expressive rendering. We propose a *generalized model* for plants and trees. This model maintains an approximated representation that keeps the tree's visual appearance at a higher level of abstraction.

Using a generalized model has several advantages. We can correctly represent the branching structure of the tree by using contour lines for each branch. We can define an abstract model for the leaves that supports different types of leaves and different ways of rendering them. When rendering the leaves we can apply different illumina-

tion models. We use our generalized model to obtain the information needed to generate the contours and render them using a suitable illumination model. We describe our generalized model for both branches and leaves.

3.1 Modeling the Branches

Branches are typically modeled using separate geometric primitives like truncated cones, generalized cylinders and polygonal meshes. In order to avoid discontinuity and visibility problems at the branch joints, we use a single polygonal model for the entire branching structure of the tree [15]. This is illustrated in Fig. 2. Using a single model we can easily apply geometric simplification methods to build a multiresolution representation of the branches with different LODs.

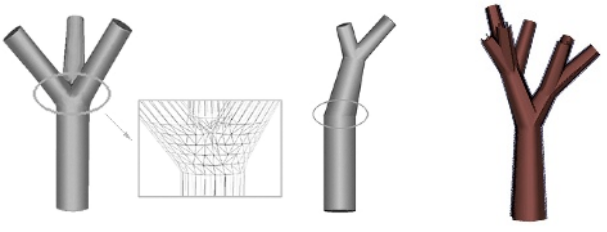

Fig. 2. Using a single polygonal mesh to represent all the branches of a tree

3.2 Modeling the Leaves

We propose a generalized model for the leaves that preserves the visual appearance of their geometric representation. To build a model we start with an RL-system and follow these steps: (i) we compute the convex hulls of each of the branches and its leaves, (ii) we compute oriented bounding boxes for the convex hulls, and (iii) we replace the bounding boxes with substitution shapes for final rendering.

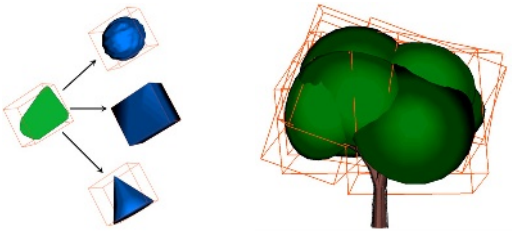

Fig. 3. Left, a convex hull and its bounding box can be replaced by one of several substitution shapes. Right, our sample tree rendered using spheres as substitution shapes

For step (i) we assume that the leaves are made of strokes, whose convex hull can be easily computed. For each branch and its sub-branches and leaves we compute the

convex hull using an algorithm due to O'Rourke [16]. In step (ii) we compute an oriented bounding box for each convex hull. An oriented bounding box is the bounding box with minimal volume that encloses the convex hull. In step (iii) of our algorithm we replace each bounding box by a generic *substitution shape*. A substitution shape can be any object that we want to render in place of a branch and its sub-branches and leaves. Fig. 3 left shows three examples. Once the substitution shapes have been generated we can render the tree (see Fig. 3 right).

4 Rendering

We compute the silhouettes using an enhanced version of the algorithm by Raskar and Cohen [18]. We apply an illumination model like the one proposed by Hamlaoui [19]. The original algorithm by Raskar and Cohen computes the wire-frame silhouette of a polygonal mesh. We modified this algorithm to draw the silhouette using texture mapped polygons. We start by computing all the polygons that belong to the silhouette. Then, we replace each of the edges of those polygons by a new polygon whose scale depends on the normal to the edge. Finally, we texture map and render those silhouette polygons.

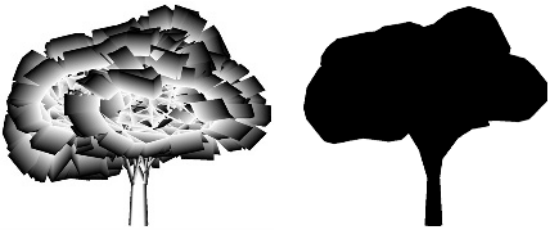

Fig. 4. Left, polygonal model generated from the silhouette polygons of our sample tree. Right, mask to remove interior edge polygons

The main drawback of this algorithm is that we need to remove those polygons that have been generated for interior hidden edges of the original polygon. A simple solution to this problem draws the mesh during a second pass, once the hidden polygons have been removed.

This method yields an analytical description of the silhouette, which can be used to create and texture map new polygons. The method is fast enough to allow interactive frame rates. Fig. 4 left shows the polygons generated from the silhouette of our sample tree. In order to remove the polygons associated to the interior edges of the silhouette we generate a mask for the tree (see Fig. 4 right). Then, we choose a texture for the silhouette polygons and we render the final image by composing it with a suitable background (see Fig. 5).

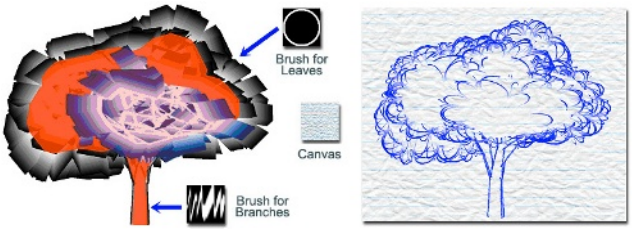

Fig. 5. Composing the final image for silhouette/based rendering

We use a modified version of Hamlaoui's illumination model [19]. Our version supports both flat and gradient shading. The idea is to apply a 1D texture to the polygons of the silhouette. The 1D texture contains a gradation of the same color. Given a vertex normal and its view and light vectors we apply an illumination model and determine which texture coordinate to use in order to obtain the desired shading. The texture element stored for that coordinate determines the final color of the vertex.

The final image is the combination of the results of a two-pass algorithm. In the first pass the algorithm computes the silhouette of the model. In the second pass, the silhouette is shaded using the appropriate illumination, color and tone. The result is a toon-shaded tree like the one shown in Fig. 6.

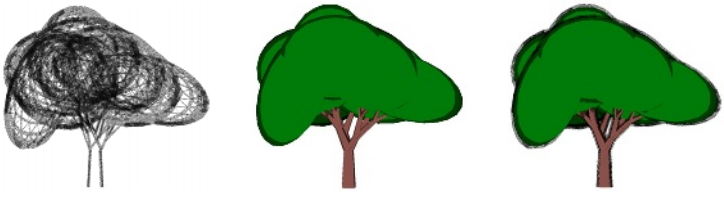

Fig. 6. Left, 1st pass - silhouette. Middle, 2nd pass - shading. Right, combined result

We demonstrate our rendering algorithm by running it on three different graphics cards, a Creative GeForce 2 GTS Pro 64 Mb, an ATI Radeon 9200 128 Mb, and an nVidia GeForce FX 5200 128 Mb. We render our sample tree at five different LODs. We use two rendering algorithms: silhouette rendering (see Fig. 7 up) and toon shading (see Fig.7 down). Our silhouette rendering algorithm runs at interactive rates (see Table 1) making it suitable for interactive walkthroughs. Our toon shading algorithm runs as fast, as shown in Table 1.

5 Conclusions

In this paper, we introduce the generalized model for representing trees in an abstract way suitable for expressive rendering. Our model stores a single polygonal mesh and preserves the visual appearance of any given tree. We can abstract the leaves' representation to obtain different leaf rendering styles. Our leaf representation supports multiple illumination models.

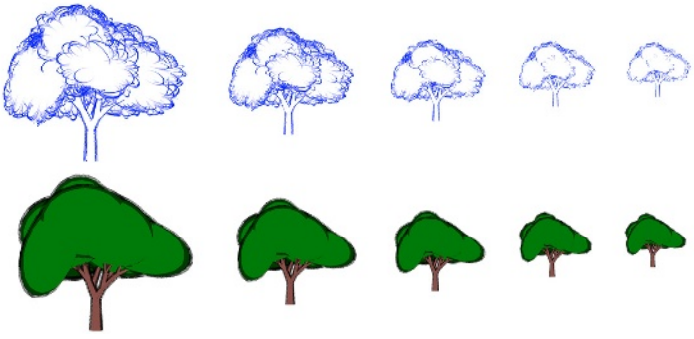

Fig. 7. Our sample tree rendered at different LODs using silhouette rendering (up) and toon shading (down)

Table 1. Frame rates achieved by our rendering algorithms

	Silhouette Rendering			Toon Shading		
	Creative	**ATI**	**nVidia**	**Creative**	**ATI**	**nVidia**
Original	35	48	60	58	60	98
LOD 1	45	58	75	60	62	99
LOD 2	65	78	105	100	102	127
LOD 3	105	95	140	125	126	142
LOD 4	125	128	155	133	135	155

Fig. 8. Other results obtained with different rendering styles

References

1. P. Prusinkiewicz and A. Lindenmayer, The algorithmic beauty of plants, Ed. Springer-Verlag, 1990.
2. P. Prusinkiewicz, M. James, and M. Mech, "Synthetic Topiary," Computer Graphics, pp 351-358, 1994.
3. T. Strothotte and S. Schlechtweg, Non-photorealistic computer graphics : modeling, rendering, and animation. San Francisco, CA, Morgan Kaufmann, 2002.
4. B. J. Meier, "Painterly rendering for animation", Proceedings of SIGGRAPH 96, pp 477-484, Agosto 1996. New Orleans, Louisiana.
5. M. Webb, E. Praun, A. Finkelstein, and H. Hoppe, "Fine Control in Hardware Hatching", Proceedings of SIGGRAPH 02, 2002.

6. C. I. Yessios, "Computer drafting of stones, wood, plant and ground materials." Proceedings of SIGGRAPH'79 (Computer Graphics), pp 190-198, 1979.
7. T. T. Sasada, "Drawing Natural Scenery by Computer Graphics.," Computer-Aided Design, vol. 19, pp 212-218, 1987.
8. M. A. Kowalski, L. Markosian, J. D. Northrup, L. D. Bourdev, R. Barzel, L. S. Holden, and J. F. Hughes, "Art-Based Rendering of Fur, Grass ant Trees", Proceedings of SIGGRAPH 99, pp 433-438, Agosto 1999. Los Angeles, California.
9. L. Markosian, B. J. Meier, M. A. Kowalski, L. S. Holden, J. D. Northrup, and J. F. Hughes, "Art-based Rendering with Continuous Levels of Detail", in NPAR 2000, Annecy, France, 2000.
10. O. Deussen and T. Strothotte, "Computer-generated pen-and-ink illustration of trees", Proceedings of SIGGRAPH 2000, pp 13-18, Julio 2000.
11. F. Di Fiore, W. Van Haevre, and F. Van Reeth, "Rendering Artistic and Believable Trees for Cartoon Animation", in CGI2003, 2003.
12. C. Campos, R. Quirós, J. Huerta, M. Chover, J. Lluch, and R. Vivó, "Non Photorealistic Rendering of Plants and Trees", in International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging., Grecia, 2001.
13. C. Campos, E. Camahort, R. Quirós, J. Huerta, and I. Remolar, "Acceleration Techniques for Non-Photorealistic Rendering of Trees", Iberoamerican Symposium on Computer Graphics, Guimaraes, Portugal., 2002.
14. J. Lluch, M. J. Vicent, R. Vivó, and R. Quirós, "GREEN: A new tool for modelling natural elements", in WSCG'2000 International Conference on Computer Graphics and Visualization, Plzen, Checz Republic, 2000.
15. J. Lluch, M. J. Vicent, C. Monserrat, and S. Fernández, "The modeling of branched structures using a single polygonal mesh," IAESTED Visualization, Imaging, and Image Processing, 2001.
16. J. O´Rourke, Computational Geometry in C, Cambridge University Press, 1998.
17. G. Barequet and S. Har-Peled, "Efficiently Approximating the Minimum-Volume Bounding Box of a Point Set in 3D," Proccedings 10th ACM-SIAM Symposium on Discrete Algorithms, 1999.
18. R. Raskar and M. Cohen, "Image Precision Silhouette Edge", In Proc. 1999 ACM Symp. on Interactive 3D Graphics, 1999.
19. S. Hamlaoui, “Cel-Shading”, GameDev.net., 2001
http://www.gamedev.net/reference/ /programming/features/celshading