Modeling and Rendering of Individual Feathers

CRISTIANO G. FRANCO, MARCELO WALTER

PIPCA – Mestrado em Computação Aplicada
Universidade do Vale do Rio dos Sinos, São Leopoldo, RS
{cfranco,marcelow}@exatas.unisinos.br

Abstract. We present a model for geometric modeling and rendering of individual feathers for computer graphics purposes. The model represents the feather structure with a collection of parameterized Bézier curves. This parametrization allows easy generation of the existent types of feather structures with biologically-motivated parameters. Once the feather structure is defined, the feathers can be rendered either in a non-photorealistic rendering style or using texture mapping to achieve a more natural look.

1 Introduction

The modeling of natural phenomena in computer graphics has been addressing phenomena from all kingdoms: mineral, vegetable, and animal. It is interesting to note that the development of this field of research in computer graphics is mimicking the complexity of Nature itself. We can already model satisfactorily many phenomena in the mineral [12, 10] and vegetable kingdoms [9, 5, 14]. The research frontier is certainly in the animal kingdom, with countless possible avenues left for research – mainly humans and animals. Among these we mention the modeling of birds, and particularly feathers. Although feathers have exquisite structures and colors, they have been, until recently, completely neglected by the computer graphics community as a research topic.

In this paper we propose a biologically inspired model for modeling the structure of feathers and also their rendering. We have chosen this biological approach – as opposed to a more algorithmically approach – since biologically motivated models have the potential to deliver more realistic renderings [11]. In previous work [6] we have presented a simple model for generating only the structure of feathers. Here we present our extended model and introduce an effective way for addressing the rendering of the feathers, taking advantage of our Bézier structure.

2 Previous Work

In the context of modeling natural phenomena, when searching for inspiration to model feathers, one would have to look necessarily into previous approaches modeling branching structures such as L-systems [13]. L-systems and all its extensions have been very successful at modeling trees and vegetation in general, and have recently been used to model the structure of feathers [2]. Another possibility would be to model feathers as particle systems [15]. Particle systems have been used for modeling fuzzy phenomena, e.g., fire, explosions, and even forests [16]. While both approaches – L-systems and particle systems – would have potential to be explored for feather modeling, we decided for designing and implementing a specific model targeted for feathers, mainly because we believe that in doing so we would have better chances of succeeding.

The only published model targeted specifically for feathers was presented by Wen-Kai Dai *et al.* [3] in 1995. In their approach, they model the structure of feathers as line segments branching from a main structure. This approach was inspired on other branching objects, such as trees and plants [1, 16]. The overall structure can be controlled by user-defined functions acting on the parameters. The need to define these functions introduces a heavy burden for the user. The visual aspect of their feathers was implemented through texture mapping, where the textures were computed from simulations of dynamical systems. A main limitation of this work is that it only models one type of feathers (from *Galliformes*).

Recently, yet-to-be published papers have addressed the modeling and rendering of feathers [2, 18]. The work by Chen and colleagues [2] uses parametric L-systems for the modeling of the feathers and texture mapping and customized Bidirectional Texture Functions [4] for the rendering. The results are visually impressive and they also developed a technique to arrange a collection of feathers on the bird's polygonal model.

Our work shares many similarities with the model by Streit and Heidrich [18], but both have been developed independently. In Streit and Heidrich's model, the feathers are modeled as a collection of Bézier curves, as in our case. The overall shape of the individual feathers is achieved by the user specifying key barbs from which the other barbs are derived by interpolation. The rendering uses texture mapping to add color to the barbs. While the idea of interpolating key barbs is powerful for some cases, it demands the user specification of each key, and for some feathers and bird coats this task could be time consuming and boring for the user. Our model has a more direct way of specifying the overall feather shape, leaving for the system the automatic generation of barbs based on a set of parameters. It is fair to say that in general terms their model has a more ambitious goal than ours since they are able to model also the growth of feathers and the modeling of the bird's coat.

3 Feather Basics

Since our model derives its inspiration from real feathers, we briefly review in this section basic information on the biology of feathers. Feathers cover the body of birds, playing an important role during flight. They also have many important properties, among them thermal properties, responsible for maintaining the bird's body temperature.

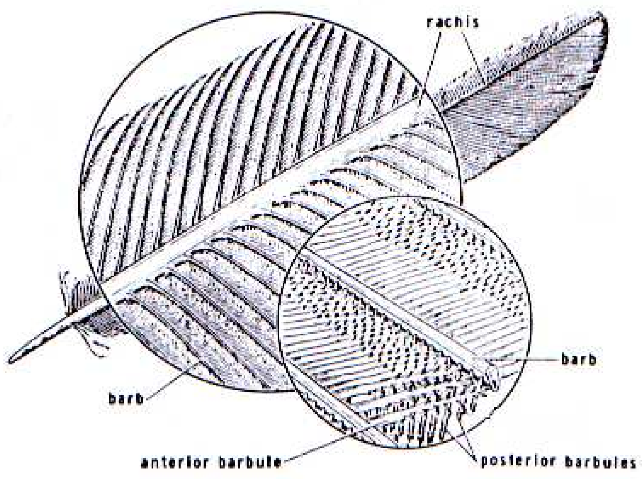

Figure 1: Structure of a typical feather (from [19]).

Feathers are a type of branching structure, flexible and yet strong [7]. They present a main rigid structure called *calamus* at the base (with no branching structures) and the *rachis* where the main body of the feather develops [8] (see Figure 1). From the rachis a variable number of *barbs* are originated. The collection of barbs at each side of the feather's body is known as *vanes*. Each barb is built from two sets of interconnected *barbules*: the anterior and the posterior barbules. For some feathers each barbule has in turn microscopic *barbicles* (not labeled in the figure), structures with small hooks that connect the anterior barbule of one barb to the posterior barbule of the next barb. This connection helps maintaining the feather overall shape.

There are 5 types of feathers and the main 4 types are illustrated in Figure 2. The most familiar type is the *contour* feather (Figure 2(b)). The semiplume (Figure 2(d)) has a structure between the contour and the plume. The plume type (Figure 2(c)) is soft and the length of the barbs is typically longer than the length of the rachis itself. Finally, the filoplume (Figure 2(a)) and the bristles (not shown in the figure) are very small specialized feathers.

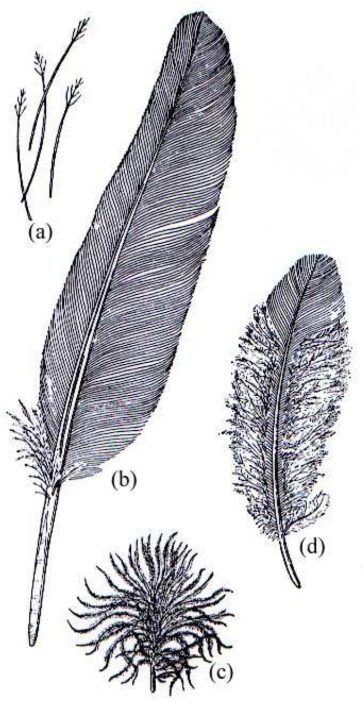

Figure 2: Types of feathers (hand drawn): (a) filoplume, (b) contour, (c) plume and (d) semiplume.

4 The Proposed Model

4.1 Overview of the Modeling Process

To model a single feather, the user initially defines a cubic Bézier, which represents the rachis, and two Bézier curves with five control points, that define the boundaries of the overall feather structure (the vanes). From the rachis we generate a variable number of barbs, controlled by the parameters explained below. Each barb is itself a Bézier segment with 4 control points.

4.2 Modeling the barbs

In order to construct the individual barbs we follow a well-defined procedure. As we can see in Figure 3, for each Bézier curve representing a barb, we will have four control points: P_0, P_1, P_2, and P_3. Even though we are using cubic segments, the resulting Bézier in practice has enough flexibility to represent a wide range of possible barb shapes. We initially assume that P_0 and P_1 are the same point on the rachis and that P_2 and P_3 are also a single point located on the limit of the vane defined by the user. The procedure will move the points P_1 and P_2 away from their initial position.

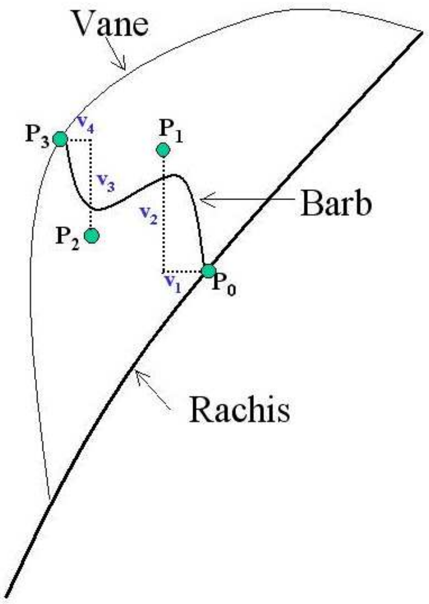

Figure 3: Constructing the structure of a single barb.

In Figure 3 we illustrate this process. First we compute the distance d between points P_0 and P_3. In order to move the point P_1, we draw two random values v_1 and v_2, following an uniform distribution, with v_1 in the interval $[-dF_b, 0]$, and v_2 in $[-dF_b, dF_b]$. F_b controls the amount of variation and will be more explained in the next section. The final position of P_1 will be given by: $P_{1x} = P_{0x} + v_1$, and $P_{1y} = P_{0y} + v_2$. In an analogous way, for the point P_2 we draw two values v_3 and v_4, with v_3 in $[0, F_b]$ and v_4 in $[-dF_b, dF_b]$. The final position of P_2 will be given by: $P_{2x} = P_{3x} + v_4$, and $P_{2y} = P_{3y} + v_3$.

This method of applying random values to the points was chosen because it is simple and visually efficient. At the same time, the other parameters will also influence this process. For instance, F_{pv} may change the distance d between points P_0 and P_3, and also parameters such as U_c and U_f will determine whether or not the random values will be generated new for each barb or if they will be kept the same for all barbs. We explain next the exact roles of F_{pv}, U_c, U_f, and other parameters.

4.3 Parameters

The study of biological formation of feather structures [19, 8] and research in previous work [3] has allowed the identification of a minimum set of parameters which can be manipulated such that we can create the structure of a feather as we wish. Besides these parameters, we have others that are not strictly related to biology and were added to facilitate the process of building different feathers structures. Our model has 7 main parameters summarized in Table 1.

Parameter	Range Values	Description
N_b	[0, 5000]	Number of barbs
F_{pv}	$[0, 1]$	Variation on length of barbs
F_b	$[0, 1]$	Variation on form of barbs
S_v	$T \vee F$	Symmetry of vanes
U_f	$T \vee F$	Uniformity of barbs (form)
U_c	$T \vee F$	Uniformity of barbs (length)
I_s	$[0, 1]$	Start of second segment

Table 1: The parameters of the model.

This set of parameters allows real time generation of many feather structures, from contour feathers to semiplumes, plumes, and even filoplumes. The meaning and role of each parameter is explained below:

- Number of Barbs - N_b
  This parameter is intuitive and determines the number of barbs that the feather will have in each vane. In practice we have used up to 2000 barbs.

- Variation on length of barbs - F_{pv}
  This parameter allow us to randomly change the length of each barb individually as they are by default limited to the boundary of vanes that were defined by the user. F_{pv} will determine how much this limit will vary in percentage. For each barb, one random value between $-F_{pv}$ and $+F_{pv}$ will be generated, and this value will be used to rescale that barb's length. Visually, the effect of this parameter is to change the position of point P_3 (in Figure 3) away either positively or negatively from the limiting Bézier defining the vane. We illustrate in Figure 4 the effect of changing F_{pv} on the resulting feather.

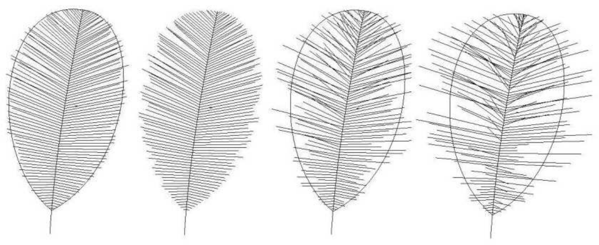

Figure 4: Variation on length of barbs: (a) $F_{pv} = 0.1$ (with border), (b) $F_{pv} = 0.1$ (no border), (c) $F_{pv} = 0.4$, (d) $F_{pv} = 0.7$.

- Variation on form of barbs - F_b
  This parameter affects directly the form of each barb. The values v_1, v_2, v_3 and v_4, which determine the points P_1 and P_2, will be random values affected by F_b which is used to scale the distance d between P_0

and P_3. In Figure 5, we can see results of different values for F_b.

- Symmetry of vanes - S_v
  This parameter is used to make the left vane identical or not to the right vane. If S_v is set to TRUE then the form and the length of the barbs will be the same in both vanes. As a boolean value, if this parameter is set to FALSE, different values will be generated from the F_b and F_{pv} parameters, and the barbs in the left vane will be different from the barbs in the right one.

- Uniformity of Barbs (shape) - U_f
  Some types of feathers, in special the contour feathers, have barbs with a regular form, i.e., all barbs have almost the same shape. For simulation of this effect the parameter U_f is used. As we explained before, in order to build the Bézier curve that represents a barb four random values are generated. When the boolean parameter U_f is set to TRUE, the first four generated values are kept, so that all barbs will have the same relative form.

- Uniformity of Barbs (length) - U_c
  This parameter is used to maintain or not all barbs with the same length. In the implementation, if this parameter is set to TRUE we do not pick a different F_{pv} each time.

- Start of second segment - I_s
  As we saw in Section 3, there are different kinds of feathers and each one has several different structures. Thus, some feathers may have regular structures, while others may have irregular structures, where the barbs are not connected. Between these two main types, there are feathers whose structure is regular in a region and irregular in another one. We use the parameter I_s to simulate this effect. Using this parameter, it is possible to divide the rachis into two segments, at the point defined by I_s. Thus, it is possible to select different values to the parameters for each segment. For example, it is possible to define the barbs uniformly in the first segment and non-uniformly in the second segment.

With these parameters we are able to build a wide range of different feather structures. In Figures 6 and 7 we show three examples of synthesized feathers with no texture attached, just to emphasize possible structures.

4.4 Rendering

Once the structure of a single feather is built, we can render it in a non-photorealistic way (as exemplified in Figure 6) or we can apply a texture and render with the texture. For

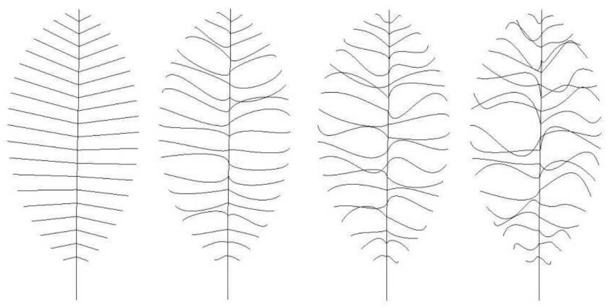

Figure 5: Variation on form of barbs: (a) $F_b = 0$, (b) $F_b = 0.25$, (c) $F_b = 0.5$, (d)$F_b = 0.75$.

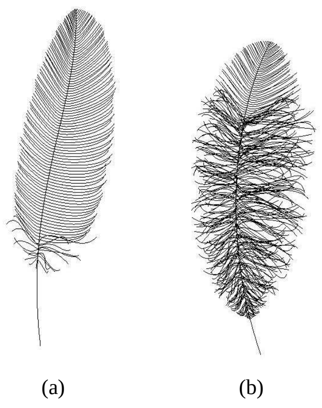

Figure 6: Examples of results rendered with non-photorealistic style. Compare these with feathers in Figure 2 (b) and (d).

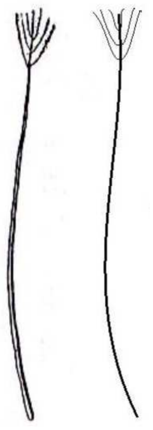

Figure 7: Synthesized and hand draw filoplume. The feather on the left is the drawing.

each feather we maintain the set of control points which define the rachis and the barbs at each vane (left and right). The sampling of the Bézier curves is user-controlled and it allows us to generate the feathers at multiple multiresolution levels, from coarse (very few sampling points) to fine (many sampling points). This could be used, for instance, to fine-tune the rendering according to the distance the camera is from the feather. The results in this paper were all generated sampling the barb Bézier curves with 30 points and connecting these points with line segments.

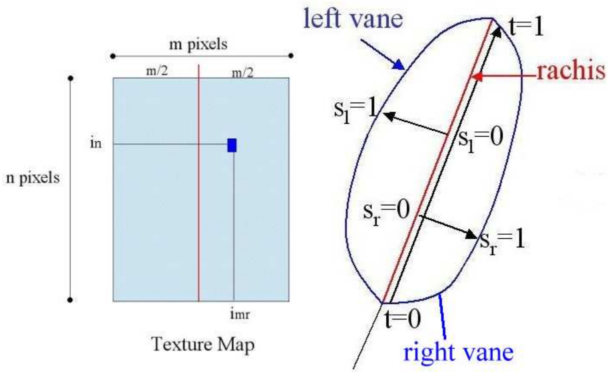

Figure 8: Texture mapping a feather.

The texture map can be either images of real feathers or artificially painted images. The best results are, of course, with real feather images. The way our feather is constructed allows us to easily compute texture map coordinates, since the Bézier curves (rachis and barbs) are already parameterized in the $[0, 1]$ interval. When rendering the individual barbs, we compute the (s, t) texture coordinates as follows: the s coordinate will vary from 0 at the rachis to 1 at the outer boundary of each of the vanes; the t coordinate will vary from 0 at the base of the feather to 1 at its tip.

We split the texture map in the middle into two parts, such that the left part will be mapped to the feather's left vane whereas the right part will be mapped to feather's right vane (see Figure 8). Given a texture image with resolution mxn and texture coordinates s_l (for the left vane), s_r (for the right vane) and t for the rachis, we compute the index (i_{ml}, i_n) or (i_{mr}, i_n) to the texture map as follows: $i_n = \lfloor t.n \rfloor$ $i_{ml} = \lfloor (1 - s_l)m/2 \rfloor$ $i_{mr} = \lfloor (s_r.m/2) + m/2 \rfloor$.

5 Results

Our system is implemented in C++ using OpenGL [20]. All results are rendered in real time on a Pentium II 400Mhz with a GeForceII graphics card and 128Mb of RAM. Table 2 summarizes the parameters used for each result presented in the paper.

In general we were able to render feathers visually similar to the real ones (Figures 9 and 10). The peacock re-

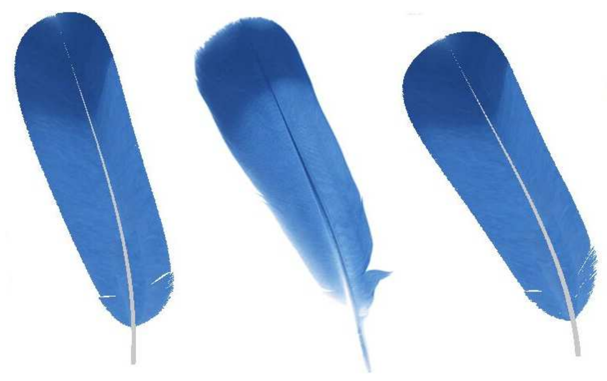

Figure 9: One example of our results. The feather in the middle is the real feather.

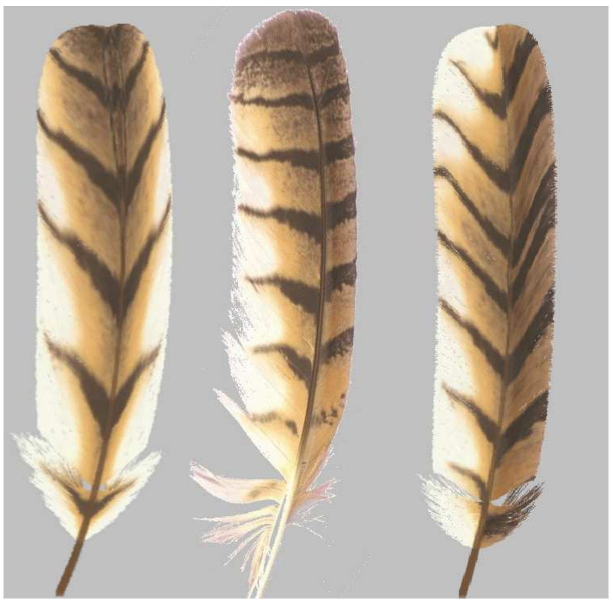

Figure 10: Hawk feather. The feather in the middle is the real feather.

Figure	N_b	F_{pv}	F_b	S_v	U_f	U_c	I_s
Figure 6 (a)	10	0.6	0.8	T	F	F	0.25
	80	0.04	0.4	T	T	F	-
Figure 6 (b)	80	0.3	0.5	F	F	F	0.8
	20	0.01	0.4	T	T	F	-
Filoplume, Figure 7	0	0	0	F	F	F	0.85
	4	0.15	0.3	F	F	F	-
Blue Feather	200	0.1	0.4	T	T	F	0.2
	2000	0.01	0.2	T	T	F	-
Hawk Feather	200	0.2	0.3	F	F	F	0.2
	2000	0.03	0.2	T	T	F	-
Peacock Feather	300	0.4	0.5	F	F	F	0.4
	600	0.03	0.3	T	T	F	-

Table 2: Values of parameters for the results. The second line in each entry presents the parameters for the second segment of the feathers.

sult (Figure 11) illustrates an artistic possibility of using our system. In general, the more regular feathers (contour) will have small values for F_{pv} and F_b whereas semiplumes and plumes will demand more irregularity and therefore larger values for these parameters. In comparison with the approach presented by Dai [3], ours has approximately the same number of parameters but they are defined and used in a more intuitive way. Our results are visually similar to the results presented by Streit and Heidrich's model [18] although our method presents a more direct way of specifying the overall shape of the feather with the boundary left and right Bézier curves defining the vanes.

6 Conclusions

We have presented in this paper a biologically motivated model for modeling and rendering individual feathers for computer graphics use. The model divides the task of modeling the feathers into two parts: modeling the structure of feathers and the rendering. The user defines the main structure with three Bézier curves that define the feather overall shape. From this shape the system constructs the individual barbs as Bézier curves, based on a set of user-defined parameters. The rendering step can either render the result in a non-photorealistic style or use texture mapping to add colours to the individual barbs. Each barb is rendered with a collection of line segments connecting the sampling points on the curve. Although we have not explored this possibility yet, we can use this feature to model the feathers at multiple resolutions.

Our system allows a great deal of user control and this can be both good and bad. We plan to include a library with the most common feather shapes and also textures that the user could load and play with, while generating the feathers. Possible interesting and more ambitious extensions to this work include the investigation of an illumination model targeted for feathers [17] and the ultimate goal of covering a bird's body.

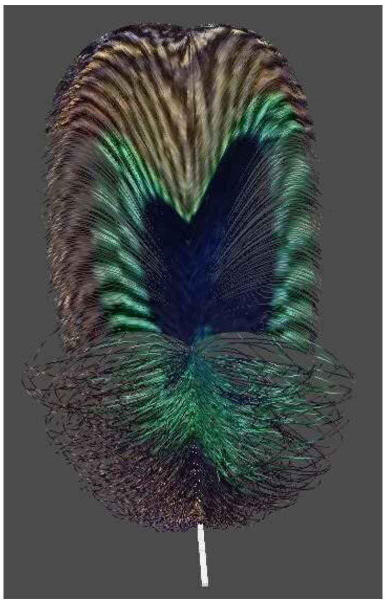
Figure 11: An example of an artistic feather resembling a peacock feather.

References

[1] J. Bloomenthal. Modeling the mighty maple. In *SIGGRAPH 1985 Conference Proceedings*, pages 305–310, 1985.

[2] Y. Chen, Y. Xu, B. Guo, and H.Y. Shum. Modeling and rendering of realistic feathers. In *Proceedings of ACM SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, page to appear. ACM Press / ACM SIGGRAPH, July 2002. ISBN 1-58113-292-1.

[3] W. Dai, Z. Shih, and Z. Chang. Synthesizing feather textures in galiformes. In *EUROGRAPHICS'95*, pages 407–420, August 1995.

[4] K.J. Dana, B. Ginneken, S.K. Nayar, and J.J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999. ISSN 0730-0301.

[5] O. Deussen, P. M. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 275–286, Orlando, Florida, July 1998. ACM SIGGRAPH / Addison Wesley. ISBN 0-89791-999-8.

[6] C.G. Franco and M. Walter. Modeling the structure of feathers. In *SIBGRAPI 2001 - Proceedings of the XIV Brazilian Symposium on Computer Graphics and Image Processing*, pages 381–381. IEEE Press, Oct 2001.

[7] R. Freethy. *How Birds Work: A Guide to Bird Biology*. Blandford Press, 1982.

[8] O.S. Pettingill Jr. *Ornithology in Laboratory and Field*. Burgess, 1970.

[9] B. Lane and P. Prusinkiewicz. Specifying spatial distributions for multilevel models of plant communities. In *Proceedings of Graphics Interface 2002*, pages 128–137, May 2002.

[10] G.S.P. Miller. The defi nition and rendering of terrain maps. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, volume 20, pages 39–48, Dallas, Texas, August 1986.

[11] P. Prusinkiewicz. Modeling and visualization of biological structures. In *Proceedings of Graphics Interface 1993*, pages 128–137, May 1993.

[12] P. Prusinkiewicz and M. Hammel. A fractal model of mountains and rivers. In *Graphics Interface '93*, pages 174–180, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

[13] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[14] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 289–300. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[15] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, April 1983.

[16] W. T. Reeves. Aproximate and probabilistic algorithms for shading and rendering structured particle systems. In *SIGGRAPH 1985 Conference Proceedings*, pages 313–320, 1985.

[17] M. Schramm, J. Gondek, and G. Meyer. Light scattering simulations using complex subsurface models. In *Graphics Interface '97*, pages 56–67. Canadian Information Processing Society / Canadian Human-Computer Communications Society, May 1997. ISBN 0-9695338-6-1 I.

[18] L. Streit and W. Heidrich. A biologically-parameterized feather model. *Computer Graphics Forum (Eurographics 2002)*, 21(3):to appear, September 2002.

[19] J. Van Tyne and A.J. Berger. *Fundamentals of Ornithology*. John Wiley & Sons, 1976.

[20] M. Woo et al. *OpenGL Programming Guide: The Official Guide to OpenGL*. Addison-Wesley, 1999.