Real-time Fur Using GPU-based Raycasting

Martin Berger
Supervised by: Petr Kmoch

Faculty of Mathematics and Physics
Charles University
Prague / Czech Republic

Abstract

Rendering fur is an important area of computer graphics, because visually convincing fur is essential for realism of games and computer generated imagery. Our paper presents a novel technique for fur rendering based on a previously published method for grass rendering. The technique works by tracing a ray at each pixel through the fur volume with implicitly defined slices mapped with fur texture. A detailed analysis with performace tests, evaluation of applicability and comparison with another state-of-the-art technique is presented.

Keywords: Photorealistic Rendering, Programmable GPU, Fur, Volume Rendering

1 Introduction

One of the main goals of 3D graphics for many decades has been a convincing simulation and rendering of characters - humans, animals or even alien creatures. This task covers many areas of computer graphics, including model animation, facial animation, lighting models of skin and clothing and also rendering of fur or hair.

Realtime rendering of fur is an interesting area of computer graphics research because the physical properties of real fur, especially its interaction with light, are quite complex and challenging to reproduce accurately at interactive frame rates. To illustrate the incredible computational complexity of this problem, 25% of the total render time of the film *Final Fantasy: The Spirits Within* was spent on the main character's hair, according to [10].

The target applications that need to render fur in realtime are primarily games. Animals with visually pleasing fur add realism to the environment of the game. Other fields such as CGI (Computer generated imagery), where visual realism is of utmost importance, would rather prefer some of the more accurate but non-realtime methods.

Our main results presented in this paper are:

- Analysis of the grass rendering method of Habel et al. [2] along with suggestions and implementation modifications that need to be employed to adapt it to fur rendering. We report all problems encountered and present a discussion of possible solutions, along with implementation details for some of them.
- Discussion of existing real-time fur rendering methods and their comparison to the custom method. We evaluate the new method with respect to performance, realism, ease of implementation and applicable scenarios.

2 Related work

There are basically two approaches to rendering fur: geometric modelling of individual hair strands and volume based rendering.

The main drawback of rendering individual hair strands is the number of geometric primitives needed—for example, a typical bear has millions of hair strands, making this technique impractical for realtime rendering. This is, however, the technique that is often used in non-realtime applications, like CGI films.

Volume based techniques, on the other hand, can be effectively implemented on graphics card hardware and are often used in realtime applications, mainly because they offer a good compromise between rendering speed and visual quality.

One of the first volume based approaches is by Kajiya and Kay [4]. The authors introduced volumetric textures, called texels, for approximating surfaces and their properties, and rendered them using ray-tracing. Their technique produces a very high quality fur but is too slow for realtime rendering.

The authors also proposed a rather simple, *ad hoc* lighting model for hair, which later became the basis of several more sophisticated models by Goldman [1] or Scheuermann [11]. On the other hand, Marschner et al. [7] used accurate physical measurements to create a lighting model that matches the appearance of real hair and captures many effects not reproduced by the previous methods.

Arguably the most often used realtime technique for fur, textured shells, was introduced by Lengyel in [6], and subsequently improved by Lengyel et al. in [5].

In this approach, virtual hair is simulated and sampled into a volume texture in a preprocess step. The hair can be

generated by various methods that differ in their complexity and ability to control the properties of the generated hair. One of the simplest methods that comes to mind, is to randomly place hair strands, set them straight up and vary their opacity from the root to the tip. A more flexible method is given in [6], where a particle simulation is proposed that allows generating of different hair styles.

At run-time, the volume texture containing the generated fur is rendered as a series of concentric shells, each one shifted a bit from the center along the vertex normals. These shells are rendered semi-transparent using alpha blending on the GPU. An example of a model rendered with our implementation of the textured shells technique is shown in Figure 1.

Figure 1: A model rendered with the textured shells technique (32 layers).

This method will work everywhere except at the silhouette, where the viewing angle gets very small, and the gaps between individual shells become visible. Unfortunately, the appearance near the silhouette is critical for perceiving the characteristics and structure of the fur. To address this issue, Lengyel et al. [5] proposed adding extra geometry called fins, textured with preprocessed hair texture, and rendered perpendicular to the surface at the model's edges. The opacity of the rendered fins is set to a value dependent on the viewing angle so that the fins are visible only near the silhouette.

3 Raycasting based approach to fur rendering

Grass and fur share many physical properties opening the possibility to use similar approaches to render them in real-time. Both of these natural phenomena can be characterised as a vast number of strands, which locally point approximately in the same direction. Typical difficulties with rendering any of them include aliasing errors, significant overdraw, the effects of self-shadowing and the need of primitive sorting for correct alpha blending. Due to the described similarity, it may be interesting to adapt the algorithms designed specifically for grass rendering to simulate fur and vice versa.

One such technique for rendering grass was introduced in the paper *Instant animated grass* by Habel et al. [2]. The primary goal of our paper is to render fur using a similar approach, discuss the problems that arise, and make necessary modifications to alleviate these problems.

3.1 The technique of Habel et al.

The authors propose a new approach to grass rendering. Unlike other common techniques, which often use some kind of billboarding and require new geometry to be placed into the scene, the technique of Habel et al. does not need any extra geometry and thus can be incorporated into existing material systems with little effort.

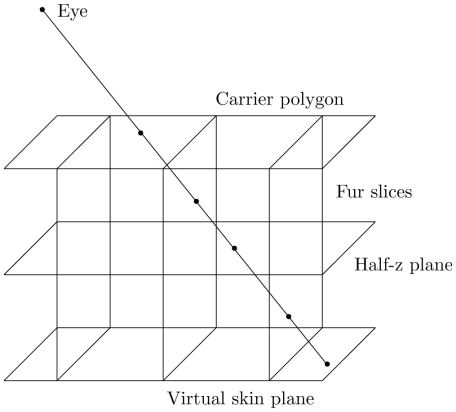

Figure 2: Visualization of ray tracing through the virtual volume containing slices with fur texture.

An outline of the technique in the context of fur rendering is as follows (see Figure 2):

- Virtual planes are defined implicitly on the model's surface. They are positioned along the u and v axes of the tangent space basis of each triangle of the model. In this way, these planes form a grid of planes perpendicular to the surface.
- In the pixel shader, a ray is traced from the viewer through the grid of planes textured with fur textures, accumulating color and opacity on its way. This method ensures, among other things, that the planes are traversed in the correct order for alpha blending.
- The ray tracing loop is terminated when the ray hits the skin plane (a virtual plane coplanar with the carrier triangle shifted by the height of virtual planes) or when a fixed maximal number of iterations is reached.
- To maintain reasonable visibility information, a depth value is calculated in the pixel shader as soon as a threshold opacity is reached.

The main assumption of this technique is that the viewer is looking at the surface mostly at grazing angles. When

viewed at perpendicular angles, the grid of planes becomes apparent. To diminish this problem, a horizontal slice, called half-z plane, is added at half of the height of the planes.

3.2 Texture map considerations

The texture with slices of fur was created manually (Figure 3), and shares the same layout with the grass slices texture described in Habel's paper. The same applies to the ground and half-z plane textures, which in our case correspond to skin and horizontal fur volume cross section, respectively. Care must be taken to make the individual fur slices unique, otherwise unwanted patterns become obvious in the rendered fur.

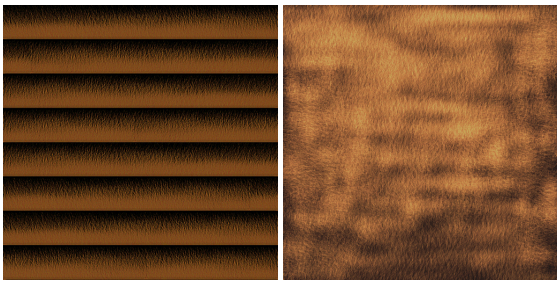

Figure 3: (a) Texture with fur slices. (b) Skin texture.

3.3 Eye direction vector

One of the first things we noticed was that the rendered fur became distorted when the camera moved close to the model's surface. When the camera was close enough for individual triangles to cover a significant area on the screen, the distortions were very noticeable and completely ruined the illusion of fur. The problem was found in the interpolation of tangent space view direction vector.

This problem can be solved by shifting the computation of the aforementioned vector from the vertex shader to the pixel shader, so that it is no longer interpolated. This modification increases the computational cost, but eliminates this problem completely.

3.4 Silhouettes

A distinctive feature of furry objects is their fuzzy appearance at the silhouette. Unfortunately, due to the nature of our method, this feature cannot be reproduced directly. The reason behind this is that the shader is executed on pixels enclosed in the projection of the model's geometry (the virtual plane space is extruded in the opposite direction of surface normals). The silhouette of this projection is composed of straight edges. Since the pixel shader can not affect pixels other than the one currently processed, the rendered fur retains straight edges at the model's silhouette. This is illustrated in Figure 4 (a).

To fix this problem, we chose to combine the basic technique with textured fins (described in Section 2). They were originally proposed to deal with a different problem (visibility of shells' structure), but as it turned out, they are able to hide the silhouette edges quite satisfactorily (see Figure 4 (b)). However, to ensure that the fins blend seamlessly with the rendered fur, the fin's texture must be similar to the texture slices used in the fur shader and also the height of the fin blades must be adjusted appropriately.

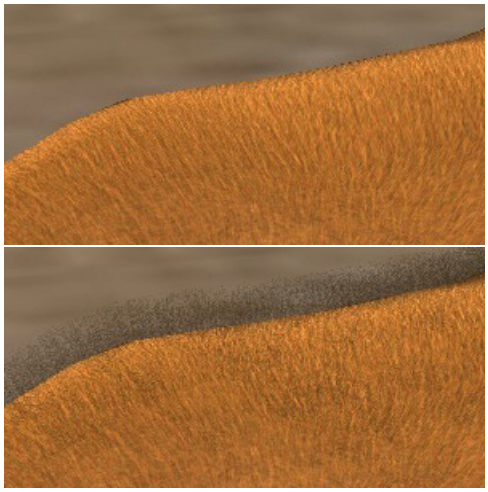

Figure 4: (a) Sharp silhouette of the original technique. (b) Original technique combined with textured fins.

The problem of silhouette appearance is not limited only to fur rendering and arises in other areas of computer graphics as well. One notable example is GPU based displacement mapping. For a detailed survey of available methods, refer to [9]. The techniques used to tackle the problem in the context of displacement mapping could provide alternative solutions to our problem. We have not investigated these possibilities further in our paper and they are left for future work.

3.5 Grid structure

The visual quality of the fur rendered with virtual planes is heavily dependent on the viewing direction. When looking at the fur at low angles, the ray tracing loop traverses many virtual planes and the visual quality is very good. However, the number of traversed planes decreases rapidly with increasing viewing angle, potentially even to zero at approximately perpendicular angles. This causes the grid structure used in the shader to become apparent, as shown in Figure 5.

The original technique was designed for rendering grass, which is rarely viewed at perpendicular angles, so this problem is not as significant there as in the case of fur rendering, where such viewing angles are very common.

We tried several approaches to address this problem. The simplest way to reduce the grid visibility is to increase the virtual plane count. Although this does not solve the

Figure 5: The grid structure is visible when looking at the surface at perpendicular angles. The model is rendered with 32 planes along each texture space axis.

problem, it makes the area with visible grid smaller. To solve this problem more thoroughly, additional modifications need to be employed.

We started with the horizontal plane mentioned in the original paper by Habel et al., positioned in the middle of the virtual plane space. It helped hide the grid on the critical places, but it also degraded the overall quality of the fur (it started having a washed-out look). Therefore we decided to modulate the color contribution of the half-z plane by a factor of $(\vec{N} \cdot \vec{V})^2$, with $\vec{N}$ being the normal vector and $\vec{V}$ the view direction vector. This factor causes the half-z plane to be visible only near the places viewed at perpendicular angles and also serves as a gradual fade-in of the visible parts.

The half-z plane does not have to be positioned exactly in half the height of the space with virtual planes. In fact, by adjusting the half-z plane position, it is possible to control its appearance to some extent. Half-z plane positioned near the top of the fur slices hides the grid but degrades the visual quality of the fur. On the contrary, rendering the half-z plane near the bottom does not help much with the grid visibility problem.

We were able to obtain quite satisfying results with dynamic half-z plane positioning. The position is given by an empiric formula we derived:

$$h = 0.9(1 - \vec{N} \cdot \vec{V}) + 0.1$$

and is calculated in the pixel shader. Note that the position is in the range $[0,1]$ with 0 being the top of the fur slices. The dot product in the formula is assumed to be properly clamped to the $[0,1]$ range. The dynamic positioning further enhances the visibility of the half-z plane in the critical places, but introduces slight visual artifacts during camera movement.

Figure 6 shows the results of the dynamic half-z positioning technique presented in this section.

3.6 Tilted virtual planes

Besides the half-z plane discussed in the previous section, we also tried some different approaches to the grid visibility problem. One of these approaches, which initially seemed very promising, is the idea of tilting the virtual

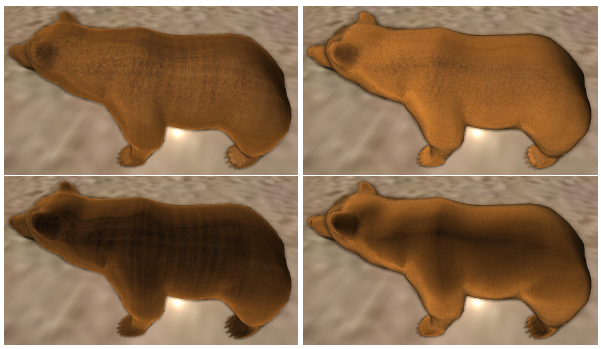

Figure 6: The effect of the half-z plane with dynamic positioning. Top left: 32 planes, half-z plane on, top right: 128 planes, half-z plane on, bottom left: 32 planes, half-z plane off, bottom right: 128 planes, half-z plane off.

planes. In the original technique, all planes are assumed to be perpendicular to the model's surface. This assumption keeps the amount of calculations inside the ray-tracing loop in the pixel shader reasonable. However, if the planes are allowed to be tilted, some parts of shader code are no longer valid, need to be generalised, and the shader complexity increases considerably.

With perpendicular planes, it is computationally inexpensive to determine the first plane intersected by the ray. Unfortunately, with tilted planes, this step is not straightforward and an exact solution would require additional calculations and non-trivial dynamic branching inside the ray tracing loop to handle the special cases that arise.

After implementing the described tilting technique, it was obvious that the visibility of the grid structure was not eliminated. Instead, this modification changed the viewing angles at which the grid was visible. Although the main problem was not solved, the tilting of the planes might still prove useful, because real fur strands rarely grow straight up.

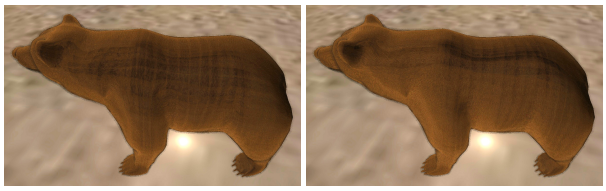

Figure 7: The effect of tilting the planes. Left: no tilting, right: planes tilted at approximately 20 degrees in both axes. Note the slant of the fur in the right image.

3.7 Problems with geometry curvature

After fixing the eye vector interpolation (see Section 3.3), the problem of severe distortions of rendered planes did not go away completely. With the camera near the surface, the rendered structure became wavy and we could

see some distortions again, though not as apparent as before.

At first, we suspected another problem with interpolation, so we tried to tessellate the mesh a lot to see whether it has any effect on the distortions. It had exactly the opposite effect from what we expected it to have. The distortions got much worse. Furthermore, we noticed that the problem was much more significant in highly curved areas.

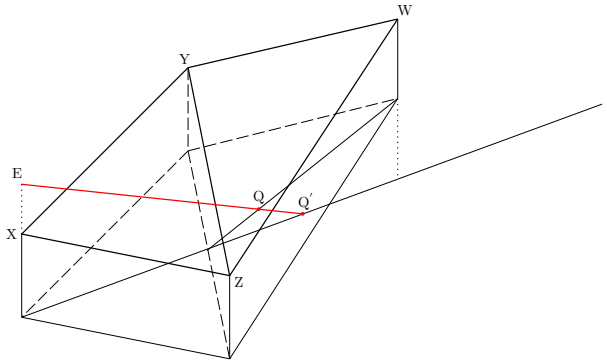

Figure 8: Schematic view of the problem with non-planar geometry. Suppose we need to find the intersection of the ray with the ground of the virtual plane volume extruded from the carrier triangle XYZ (the situation with intersecting of virtual planes is analogous). The correct intersection point is Q, since the neighbouring triangle YWZ does not lie in the same plane as the triangle XYZ. However, the ray tracing code does not account for this situation and returns the point Q' instead.

This led us to identifying the problem in the idea of extruding the volume containing the virtual planes from individual mesh triangles. During the ray tracing loop, the pixel shader has only information related to one triangle and thus cannot account for neighbouring triangles that can have different slopes. The ray traversal can eventually end up hitting planes that lie outside the volume of the original triangle (typically when the triangle is viewed at very steep angles), where the tangent space basis is no longer valid. Such a situation is depicted in Figure 8.

Unfortunately, we have to conclude that this is an inherent limitation of the original technique and there are few ways how to deal with it. This problem is not important when rendering grass, because grass usually covers some terrain with nearly coplanar individual triangles. However, fur is typically rendered on models with significant geometrical curvature.

Possible solutions to this problem include local surface approximation by quadric surfaces introduced in [8] or rendering fins on triangle boundaries as in [3].

3.8 Lighting model

To further enhance the realism of the rendered fur, we decided to implement some non-trivial lighting model. The first choice was the famous model of Kajiya and Kay [4]. The model is basically the Phong model adapted to the structure of fur (cylindrical strands). The diffuse and specular components of this model are given by:

$$\Psi_{diffuse} = k_d r \sin(T,L)$$
$$\Psi_{specular} = k_s((T \cdot L)(-T \cdot V) + \sin(T,L)\sin(T,V))^p,$$

where $k_{d,s}$ are the diffuse and specular reflection coefficients, r is the radius of the hair strands, T is the tangent vector pointing from root to tip, L is the light vector, V is the eye vector and p is the standard Phong specular exponent. For derivation of these formulas, refer to [4].

This model captures only a small part of the light scattering process of real hair strands. The biggest limitation is the fact that the model deals only with first-order light reflection. However, real hair exhibits both reflective and transmissive behavior. An accurate solution would have to consider scattering of light onto other hair strands and also onto the underlying skin. These light interactions result in physically complex visual phenomena such as secondary highlights observed on real hair (see [7] for a detailed discussion of these phenomena).

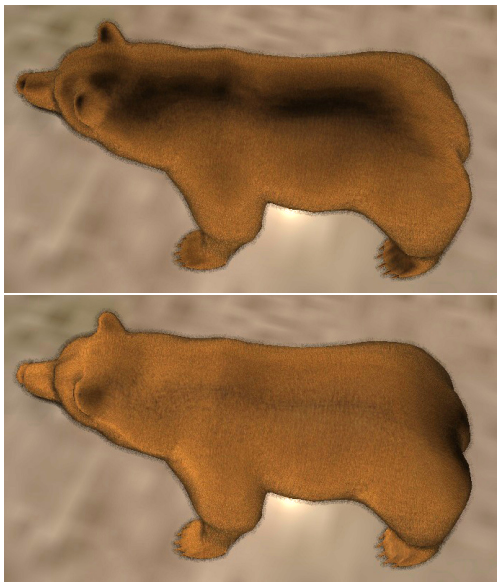

Figure 9: The lighting model (a) without and (b) with shifting of the tangent vector T. The light is positioned a bit to the left and above the bear model.

To extend the Kajiya-Kay lighting model, we added the calculation of a directional attenuation factor, as described in [1]. Adding this factor increases the directionality of the hair. Moreover, the relative reflectivity (backward scattering) and transmissivity (forward scattering) is parametrized by two factors, $\rho_{reflect}$ and $\rho_{transmit}$, that can be used to tune the produced result.

After implementing the described lighting model, the visual results were not as good as we expected. The main reason behind this is that the hair strands are positioned

straight up on the model. This is in contrast with real fur, which usually has some slant. As a quick and dirty solution, we add a tangential component to the normal vector on the surface (this vector corresponds to T in the equations). The length of the component is modulated by a sample from a noise texture to make the lighting a bit softer. The effect is shown in Figure 9.

4 Results

4.1 Performance analysis

The custom method we developed can be thought of as an alternative to the textured shells technique. Both techniques are capable of producing high quality fur, but with rather different computational requirements.

With textured shells, rendering a model with 32 layers causes the model to be submitted to the renderer 32 times. This may pose a burden both on the geometric (in case of a high polygon count model) and pixel shading stages of the rendering pipeline. However, the simplicity of the vertex and pixel shader programs makes rendering of a large number of layers feasible.

On the other hand, with the technique presented in our paper, the geometric complexity is independent of the number of virtual planes or the maximum passes of the ray-tracing loop. Also, due to the nature of the technique, the overdraw is basically zero. The complexity of this technique lies in the long pixel shader full of dynamic branching, where a ray-tracer is implemented.

The performance of this method is expected to be influenced the most by the average number of iterations of the main ray-tracing loop. There are two parameters that can affect this number: the virtual plane count and the ray-tracing loop limit. With small virtual plane counts, most of the rays intersect only a few planes before hitting the ground and breaking the loop. The maximum number of iterations is not the limiting factor in this case, because it affects only the rays that penetrate the virtual plane space at the silhouette. However, this factor becomes critical when the planes are laid out densely. The dependency of these factors is discussed in detail in the next section.

4.2 Tests

Let us examine the performance of our method in detail. For reference, we have included some tests with the textured shells technique applied to the same geometric model. In each test, the model was rotated in front of the camera for a fixed amount of time and average FPS was recorded. The individual tests were:

Tests	Shader	Resolution	planes/layers	loops
T1 - T3	Our	1600×1200	32, 64, 128	5
T4 - T6	Our	1600×1200	32, 64, 128	10
T7 - T9	Our	1600×1200	32, 64, 128	20
T10	Our	1920×1200	128	20
T11	Shells	1920×1200	32	-
T12	Shells	1600×1200	32	-
T13	Shells	1600×1200	16	-

The results of the tests, obtained on a desktop PC with Intel Core2Duo 3.0 GHz CPU, 4GB RAM and ATI Radeon HD4850 graphics adapter with 512MB of VRAM, are given in the following table:

Test	Average FPS	Test	Average FPS
T1	297.6	T8	151.6
T2	297.3	T9	139.1
T3	305.9	T10	121.7
T4	224	T11	99.9
T5	207.7	T12	108.4
T6	206.9	T13	190.6
T7	164.2		

These results clearly confirm the hypothesis that the maximum number of iterations in the ray-tracing loop is the most important parameter from the performance point of view.

Tests T9 and T12 can be chosen for visual comparison of both techniques with high quality settings. Details from the rendered images are shown in Figure 10. As can be seen from the image, the textured shells produce a more fluffy fur. The novel technique produces a fur that is optically lower but the fur structure is more detailed from a close-up look. With these settings, our method runs slightly faster (a few tens of FPS).

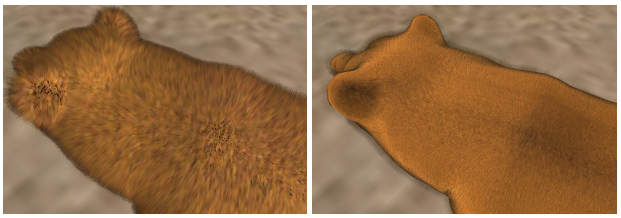

Figure 10: (a) The technique with settings from the test T12. (b) The technique with settings from the test T9.

4.3 Applicable scenarios

The question, which of the two methods to implement, is more of an artistic one. There are, however, some facts that might favor one method over the other. The fur rendering method presented in our paper is generally suitable for shorter fur types because of the behavior at the silhouette. Long fur would probably be more realistic with the textured shells technique.

The novel method integrates well into existing rendering pipelines. There are, however, some changes beyond the shader compilation and usage that need to be taken care of when implementing this technique. The most important is the generation of fins' geometry. This step is required with textured shells as well, though. The amount of work needed for authoring the textures used in the shaders is comparable for both techniques.

The target shader model of the HLSL compilation must be at least the version 3.0, as the previous versions do not support dynamic branching. This places a restriction on the hardware capable of running these shaders. This is in contrast to textured shells, whose shaders utilize only a very basic feature set enabling the use of this technique even on older hardware not capable of shader model 3.0.

Another factor, that should be taken into consideration, is the memory usage. Textured shells, being a volumetric method, consume a significant amount of memory. The proposed solution has much lower requirements, as it uses only four textures of reasonable size.

5 Conclusion

5.1 Summary

We have introduced the topic of fur rendering and discussed possible approaches along with their respective advantages and limitations. We have put emphasis on describing the textured shells technique, because it was used for comparison with the novel method.

As the main goal of the paper, we have presented a new fur rendering technique, utilizing an approach introduced in the context of grass rendering in [2] and then proceeded to a discussion of problems we encountered. We have presented solutions to most of these problems and, where possible, stated the impact of the modifications on the source code complexity and performance. We have also presented some ideas for implementation of a non-trivial lighting model. Finally, we discussed the pros and cons of the novel technique, and presented some performance tests with a comparison with the textured shells method.

5.2 Future directions

The method, as it has been described, does not include any form of LOD (Level of detail). Developing some kind of LOD could prove interesting as it would widen the range of applications of this technique. We have not investigated the possibilities in detail but we think that the parameter controlling the maximum number of ray-tracing loop iterations could be used to create some LOD scheme.

Furthermore, the last problem described in Section 3.7 needs a solution. At the end of that section, we give references to papers containing work related to similar problems. We believe that a viable solution to our problem could be derived from them.

Finally, the lighting model could be improved to account for self-shadowing and to better handle the uniform fur direction that results from perpendicular virtual planes.

6 Acknowledgements

I would like to thank Petr Kmoch for his guidance during the creation of this work.

References

[1] Dan B. Goldman. Fake fur rendering. In *Proceedings of SIGGRAPH '97*, pages 127–134, New York, NY, USA, 1997.

[2] Ralf Habel, Michael Wimmer, and Stefan Jeschke. Instant animated grass. *Journal of WSCG*, 15(1-3):123–128, January 2007.

[3] S. Jeschke, S. Mantler, and M. Wimmer. Interactive smooth and curved shell mapping. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, pages 351–360, 6 2007.

[4] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *Proceedings of SIGGRAPH '89*, pages 271–280, New York, NY, USA, 1989.

[5] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 227–232, New York, NY, USA, 2001.

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

[7] S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 780–791, New York, NY, USA, 2003. ACM.

[8] M. M. Oliveira and F. Policarpo. An efficient representation for surface details. Technical Report RP–351, UFRGS, January 2005.

[9] L. Szirmay-Kalos and T. Umenhoffer. Displacement mapping on the GPU - State of the Art. *Computer Graphics Forum*, 27(1), 2008.

[10] Scheuermann T. *Hair Rendering and Shading*. GDC presentation, 2004.

[11] Scheuermann T. Hair rendering and shading. In Wolfgang Engel, editor, *ShaderX3: Advanced Rendering with DirectX and OpenGL*, pages 239–250. Charles River Media, 2005.