Eurographics Symposium on Rendering (2004)
H. W. Jensen, A. Keller (Editors)

A Self-Shadow Algorithm for Dynamic Hair using Density Clustering

Tom Mertens[†] Jan Kautz[‡] Philippe Bekaert[†] Frank Van Reeth[†]

Expertise Centre for Digital Media
Limburgs Universitair Centrum[†]
Universitaire Campus, 3590 Diepenbeek, Belgium

CSAIL
Massachusetts Institute of Technology[‡]
Cambridge, MA

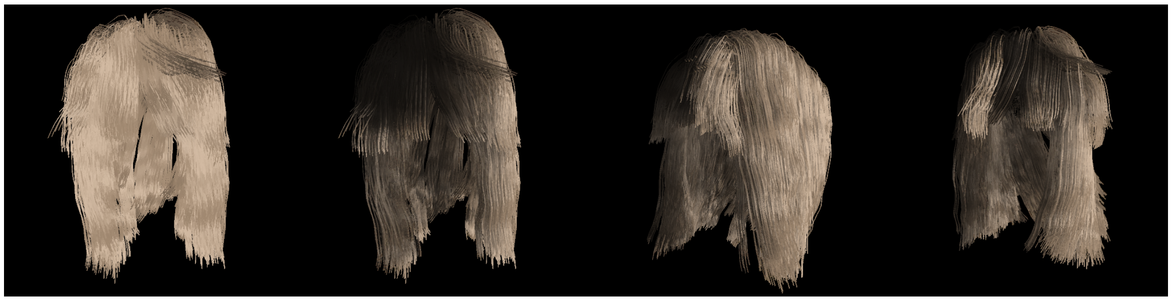

Figure 1: *Frames taken from an interactive animation (6Hz) of a hair model consisting of 100K line segments.* ***Left:*** *image without self-shadows.* ***Others:*** *images rendered with our algorithm.*

Abstract

Self-shadowing is an important factor in the appearance of hair and fur. In this paper we present a new rendering algorithm to accurately compute shadowed hair at interactive rates using graphics hardware. No constraint is imposed on the hair style, and its geometry can be dynamic.
Similar to previously presented methods, a 1D visibility function is constructed for each line of sight of the light source view. Our approach differs from other work by treating the hair geometry as a 3D density field, which is sampled on the fly using simple rasterization. The rasterized fragments are clustered, effectively estimating the density of hair along a ray. Based hereon, the visibility function is constructed. We show that realistic self-shadowing of thousands of individual dynamic hair strands can be rendered at interactive rates using consumer graphics hardware.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture

1. Introduction

Rendering hair, smoke and other semi-transparent objects is challenging, because self-shadowing has to be handled correctly. Various algorithms exist, but are either geared towards offline rendering [LV00, Sta94, KH84] or not quite interactive yet [KN01]. We present a method that allows interactive rendering of dynamic hair with dynamic lighting. While we focus on the rendering of hair in this paper, rendering other semi-transparent primitives can be handled with the same technique. In Figure 1, an example rendering using our technique is demonstrated.

Similar to deep shadow maps [LV00], we do not store a single depth at each pixel of a shadow map, but rather a fractional visibility function (or simply visibility function) that records the approximate amount of light that passes through the pixel and penetrates to each depth. This visibility function takes into account the opacities (cf. coverage) of hair strands. The partial attenuation of light passing through an object can be accurately modelled with the help of this function.

The novelty of our algorithm lies in the way the visibility function is constructed. In contrast to deep shadow maps [LV00], we do not compute the full visibility function for each shadow map pixel and compress it later — we do both at the same time. We rasterize all hair primitives into the frame buffer. During rasterization we create and update clusters of nearby primitives. These clusters are then used to construct a piecewise linear approximation to the visibility function. An object can then be shadowed, by evaluating the approximated and compressed visibility function. Our algorithm can be implemented on current graphics hardware, achieving interactive results (Figure 1). On future hardware, we expect real-time frame rates.

2. Related Work

Many techniques have been proposed for rendering shadows. Traditional shadow maps [Wil78] place a virtual camera at the light source and render the depth of all visible surfaces into a *shadow map*. A point is in shadow, if its distance to the light source is bigger than the deph-value stored in the shadow map. This algorithm cannot handle semi-transparent objects, since only a single depth value is stored.

Anti-aliased shadow maps can be rendered using *percentage-closer filtering* [RSC87]. In this case, multiple samples in a filter region are queried and the fraction of samples in shadow defines the shadow value. In theory, this can be used to render shadows from e.g. hair, but requires a large amount of samples and still often suffers from aliasing [LV00].

Crow [Cro77] proposed *shadow volumes*, where a polygonal representation of the volumes defining the shadowed regions is created. At run-time every visible point is tested if it lies inside any of these shadow volumes. It cannot be used to render semi-transparent objects such as hair, since only a binary decision can be made.

For the special case of static models but dynamic lighting, various algorithms have been proposed to render shadows for semi-transparent objects or volumes in real-time [SKS02, DHK*03, NRH03]. Our algorithm allows for dynamic models, and does not need any expensive preprocessing like these techniques.

Deep shadow maps [LV00] do not store a single depth value per shadow map pixel, but instead store fractional visibility through all possible depths. First, this (per-pixel) 1D visibility function is sampled at many locations. Then it is efficiently compressed into a piecewise linear function. Rendering shadows from this representation is simple: every visible point is intersected with the corresponding 1D visibility function. The fractional visibility is simply queried from the approximated visibility function, and then used to modulate the color at the point.

Our method is based on the same idea: construct the fractional visibility function and represent it compactly. The main difference is how we sample and compress the function. We have also decided to use a piecewise linear approximation. During sampling, we already decide on the position of the vertices of the piecewise function, which is done through k-means clustering and histogram binning of the densities. I.e., we never need to store the full 1D function before we compress it. Furthermore, our algorithm was designed to be implemented on graphics hardware, resulting in interactive performance.

Opacity shadow maps [KN01] are basically a simpler version of deep shadow maps using graphics hardware. The main difference to deep shadow maps is, that the fractional visibility function is not compressed. It is sampled regularly and stored in texture maps. Visibility queries are done using texture mapping. Regular sampling of the visibility function using graphics hardware is expensive, because it either requires sorting the primitives (and rendering them back-to-front), or slicing and re-rendering the full geometry for every sample location and (used by Kim and Neumann [KN01]). This approach requires tens of re-renderings to achieve plausible results. In contrast, our method required significantly less rendering passes. Recently, Koster et al. [KHS04] described an accelerated implementation of the opacity shadow map algorithm. In their work, real-time results are achieved, at the cost of significantly reducing the accuracy of hair representation and shadowing (empirical offsets are necessary to avoid false shadowing artifacts).

As a side note, we mention that the idea of representing hair as a density field has been proposed by Hadap et al. [HMT01] for the purpose of modeling and animation, but is not related to the approach discussed in this paper.

3. Density Clustering

Similar to the deep shadow mapping [LV00] and opacity shadow mapping [KN01], we construct a 1D visibility function along the rays of the light view. In this section we will elaborate on this.

3.1. Hair as a Density Field

We cast the problem of shadowing complex volumetric geometry into volume rendering terms. The idea is to estimate a 3D density field over the geometry of interest. This density field is analogous to the extinction coefficient σ_t, which represents the chance per unit length that light gets absorbed or scattered along a ray [Ish78]. The density field is not contructed explicitly, but rather sampled on the fly using simple rasterization. Line primitives representing hair strands are rasterized for each pixel. Due to the high depth complexity of hair, ranging up to 200 in our experiments, several rasterized fragments will map to the same pixel. Each fragment has an opacity value α associated with it, indicating the fraction of light that is blocked, i.e. the chance that a light particle get absorbed or reflected (scattered). In the case of hair, this is the fraction of the projected area of a hair strand that covers the pixel. Multiple α's occupying the same pixel are composited by multiplication. This opacity is related to the extinction coefficient σ_t as follows. Take a small ray segment of length Δs, containing N fragments. The extinction coefficient for that segment is expressed as

$$\Pi_{i=1}^{N}(1-\alpha_i) = e^{-\sigma_t \Delta s} \quad \text{iff} \quad \sigma_t = \frac{\sum_{i=1}^{N} \beta_i}{\Delta s}, \tag{1}$$

with $\beta_i = -\ln(1-\alpha_i)$. The last equation implies the density field may be computed by means of density estimation [Sil86] of β values along the ray. Once the density function is known, we can obtain the visibility function by simply integrating σ_t:

$$V(s) = e^{-\int_0^s \sigma_t dt}. \tag{2}$$

In volume rendering literature, this function is also known as the transparency, and is called transmittance function in the deep shadow map paper [LV00]. In the following sections we will define the integral function in the exponent as $T(s) = \int_0^S \sigma_t dt$, called optical depth in volume rendering.

Fast and accurate estimation of the density field σ_t will be the main challenge in our approach.

3.2. Construction of the Visibility Function

A straightforward way to estimate density is the histogram method [Sil86], which accumulates point samples of a density function in discrete bins. This essentially generates a piecewise constant approximation of the density, which results in piecewise linear optical depth after the integration of Equation 2.

Histogram bin extents can be placed either uniformly or non-uniformly along the depth. The first approach is related to opacity shadow mapping technique [KN01]: the geometry is uniformly sliced and alpha values are composited for each slice. This corresponds to accumulating β-values in uniform bins. Image quality is heavily dependent on the number of slices. Usually a high number, e.g. 80, is required to achieve reasonable results.

We wish to have a representation of the visibility which has no restriction on placement of vertices of the piecewise approximation on depth axis, i.e. the extents of the histogram bins are chosen non-uniformly. Deep shadow maps [LV00] are also based on a general piecewise linear approximation, however the construction is too complicated to be implemented in graphics hardware. The improvements over uniform histograms are discussed in section 5

To accomplish this, we cluster the fragments' depth values, and place the bins around the clusters in a way that they reflect the distribution of the intersecting geometry. A cluster represents a bundle of coherent hair geometry and causes an increase of T. This increase is proportional to the cluster's density. Assuming it is populated uniformly, T behaves

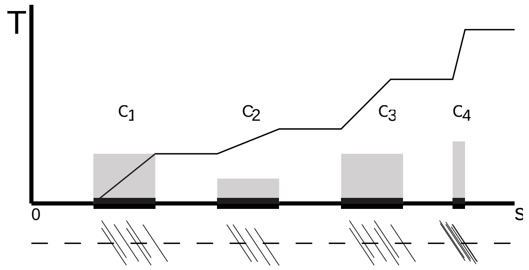

Figure 2: *Clustering of hair geometry. The dashed line represents a ray intersection hair geometry at different depths. This is obtained by rasterizing the hair. The depths are clustered, yielding C_1 to C_4, and their contribution (log of opacity) is accumulated in the corresponding cluster histogram bins. The resulting T function is obtained by integrating the histogram.*

linearly (see Figure 2). The width of the bin is related to the standard deviation σ of a cluster. It measures the spread of the fragments in the cluster, indicating the length of the increase.

More precisely, we compute our histogram as follows:

- cluster fragments while rasterizing, yielding the mean μ_j.
- rasterize again to compute standard deviation σ_j of each cluster based on its mean μ_j.
- construct K bins between $\mu_j \pm \Delta_j$, with $\Delta_j = \gamma \times \sigma_j$, and γ a constant.
- Perform histogram binning in another rasterization pass.

This procedure is depicted in Figure 2. Please note, that the geometry does not need to be sorted at all.

Again under the assumption of uniform density inside a cluster, we estimate the value of γ as:

$$\gamma = \sqrt{3}$$

This is derived from the fact that a uniform distribution of width W, has variance $\sigma^2 = \frac{1}{12}W^2$, and $W = 2\Delta_j$. Since $\gamma > 1$, two adjacent bins might overlap, and care must be taken that fragments are not accounted for twice (it can be shown that for $\gamma < 1$ bins will never overlap). We handle this exception as follows. Let $R1$ be the right hand extent of the first overlapping bin, and $L2$ the left hand extent of the second bin. We wish to avoid that $R1 > L2$, and therefore let $R1 = L2 = \frac{R1+L2}{2}$.

The remaining bins in between the cluster bins will also receive contributions, but in a much less amount. To account for these fragments we simply add their contribution to the nearest cluster.

3.3. Clustering

We opted for the k-means clustering algorithm [Llo82] because of its simplicity. First, K initial cluster positions are chosen. In an iterative fashion, the position are updated by the mean of the closest samples for each cluster, until the positions stagnate. For every iteration, the scene has to be re-rendered. We restrict the number of iterations to one. In our experience this has not compromised image quality.

Since opacities can vary significantly among fragments, we compute the mean and variance in a weighted fashion:

$$\mu = \frac{\sum_i \beta_i \times depth_i}{\sum_i \beta_i} \quad \sigma^2 = \frac{\sum_i \beta_i \times (\mu - depth_i)^2}{\sum_i \beta_i} \tag{3}$$

The efficiency of this clustering algorithm is dependent on the choice of the initial cluster positions. We choose to construct them uniformly between the nearest and furthest fragment.

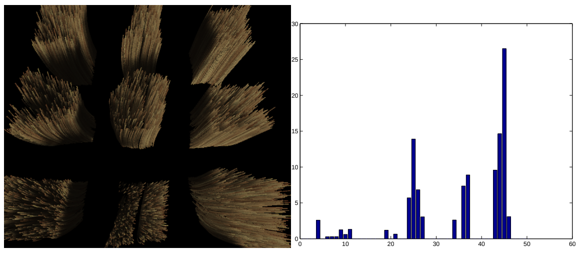

Figure 4: *Hair model used in Matlab experiment (see Figure 5). **Left:** the model consisting of several square hair bundles. **Right:** a histogram of the gathered data. The geometry was sampled by tracing a ray through the hairs, and intersects with 3 bundles.*

3.4. Filtering

Filtering of the geometry primitives is an important aspect for the case of hair, since strands usually cover only a fraction of a pixel. If not, severe aliasing artifacts will be visible in the shadows. Instead of supersampling and post-filtering [LV00], or point splatting [KN01], we rasterize pre-filtered lines during the clustering. This is accomplished using texture mapping. Each of the line segments are rendered in a fashion similar to so-called "billboards" which are commonly used to render particles.

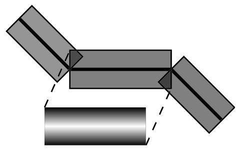

Figure 3: *Filtered hair strands: we use "billboarded" lines that have a 1D pre-filtered line texture applied.*

A rectangle is constructed in screen space such that it is aligned along the 3D transformed line segment while facing the image plane. A 1D texture contains a pre-filtered gaussian profile of the line, and is applied to the rectangle along the length of the segment. A depiction is given in figure 3. The width of the kernel controls the smoothness of the resulting shadow. Mipmapping will assure that the profile is filtered once again to match the sampling resolution of the screen.

In case of perspective projection, a line's pixel coverage decreases with depth. To account for this, the contribution of each rasterized fragment is weighted by its reciprocal depth value.

4. Implementation

The presented algorithm was implemented on an ATI Radeon 9800XT graphics board, equipped with programmable vertex and fragment shading units. We implemented the clustering for $K = 4$, since it fits best in the 4-vector format used throughout the graphics hardware pipeline. This optimally utilizes texture space and vector processing performance. The bulk of the computations of the rendering is done in fragment shaders. Except for the construction of line rectangles as described in section 3.4, which is performed in a vertex shader.

We will not go into the low level details of our implementation, since it is quite involved on current hardware. Next generation GPUs will ease this, and allow to achieve higher performance and precision. In particular, floating point blending [nVI] will have the biggest impact. Currently, to compute the sums in equation 3, we have to emulate high-precision blending on 8-bit integers using a technique that quadruples the number of rendering passes [Jam03]. Therefore, we expect our results described in section 5 to gain at least a factor 4 speedup in the future.

5. Results and Discussion

In Figure 1 several frames of an animated sequence are shown. The first frame shows the unshadowed version. It is clear that shadows have a serious impact on the realism. Figure 5 shows various other results: long hair, curly hair and even grass. For a highly detailed model of 200K line segments, we obtain an interactive frame rate of 3 Hz on an Intel P4 at 1.3Ghz. Performance scales linearly with the number of segments.

We validated the efficiency of the algorithm described in section 3 experimentally in a Matlab implementation. Also, a comparison is made with using a uniform histogram containing $2K$ bins. The results are shown in Figure 5.

In the first experiment we ran the algorithm on data gathered from an actual hair model, shown in Figure 4 and Figure 5. In the second experiment, we simulated fragment data by generating 100 random depth values between 0 and 1 using a mix of Q gaussian distributions with a random mean and variance. Two situations are shown in Figure 5: $Q = K$, and $Q > K$. It is clear that the clustering approach outperforms the uniform histogram method. The latter is prone to e.g. false shadowing, which happens when the increase starts too early. When the number of gaussians exceeds K, the method seems to break down because more actual geometric clusters get mapped to the available clusters in the representation. We note that the quality in this case is heavily dependent on the choice of initial cluster positions, which are chosen uniformly between 0 and 1. The bottom right image shows the same situation, except that initial positions have been scaled 75% towards the left. This results in a more accurate approximation near the origin, where the visibility function is more prone to visual artifacts due to the exponential fall-off (equation 2). In general, gaussians get assigned to the closest k-means cluster. This implies that those which are too close together are likely candidates to get put into a single cluster, especially when $Q > K$. Visual quality might get affected if it occurs near the origin. The choice of optimal initial clusters remains an open problem.

Comparing to Kim et al. [KN01], we achieve higher quality while display rates are one order of magnitude faster. Although the steps in our algorithm are computationally more intensive, opacity shadow mapping requires to re-render the scene significantly more than the O(K) steps in our case. The opacity shadow mapping approach is qualitatively compared to ours in figure 5. One can opbserve that the opacity shadow map is too bright in directly lit areas, and also lacks significant shadow detail there. We suspect this is mainly caused by the empirical offsets, as discussed in [KHS04]. These are indispensible to avoid false shadowing artifacts, even for a high amount of slices (256 in this case).

6. Conclusion and Future Work

We have presented a novel approach to render self-shadows for dynamic hair geometry. Each hair strand is taken into account, thus allowing every possible hair style. The key insight is that we treat hair geometry as a 3D density field, which is not reconstructed explicitly, but sampled on the fly using hardware rasterization. A clustering algorithm is used to estimate the geometry density, and used to construct a visibility function w.r.t. the light source. We have demonstrated the effectiveness of our algorithm both for rendering speed and quality. Results will dramatically improve with the advent of next generation hardware.

Several issues remain for the problem of rendering hair. First of all, aliasing still occurs when displaying the geometry in eye space. A similar method to the one describe here may be devised to also tackle this problem, possibly creating a general rendering frame work for hair, which may even be suitable to display other complex semi-transparent geometry.

As mentioned in section 5, image quality depends on the choice of initial clusters for the k-means algorithm [Llo82]. We wish to investigate how to improve this choice which is done rather ad hoc in the current implementation. Moreover, other clustering algorithms may as well apply to this problem.

Acknowledgements

This work is partly funded by the European Commission (European Regional Development Fund) and the Flemish Government (IWT).

References

[Cro77] CROW F.: Shadow Algorithms for Computer Graphics. In *Proceedings SIGGRAPH* (July 1977), pp. 242–248.

[DHK*03] DAUBERT K., HEIDRICH W., KAUTZ J., DISCHLER J.-M., SEIDEL H.-P.: Efficient Light Transport Using Precomputed Visibility. *IEEE Computer Graphics and Applications 23*, 3 (May 2003), 28–37.

[HMT01] HADAP S., MAGNENAT-THALMANN N.: Modeling Dynamic Hair as a Continuum. *Computer Graphics Forum 20*, 3 (2001), 329–338.

[Ish78] ISHIMARU A.: *Wave Propagation and Scattering in Random Media*, vol. 1. Academic Press, 1978.

[Jam03] JAMES G.: Rendering objects as thick volumes. In *ShaderX2: Shader Programming Tips and Tricks With DirectX 9*, Engel W. F., (Ed.). Wordware, 2003.

[KH84] KAJIYA J., HERZEN B. V.: Ray Tracing Volume Densities. In *Computer Graphics (Proceedings of SIGGRAPH 84)* (July 1984), vol. 18, pp. 165–174.

[KHS04] KOSTER M., HABER J., SEIDEL H.-P.: Real-time rendering of human hair using programmable graphics hardware. In *Proceedings of Computer Graphics International 2004 (CGI 2004)* (June 2004). to appear.

[KN01] KIM T.-Y., NEUMANN U.: Opacity Shadow Maps. In *Proceedings of the Eurographics Workshop on Rendering* (2001), pp. 177–182.

[Llo82] LLOYD S.: Least squares quantization in PCM. *IEEE Trans. on Information Theory 28*, 2 (1982), 127–138.

[LV00] LOKOVIC T., VEACH E.: Deep Shadow Maps. In *Siggraph 2000, Computer Graphics Proceedings* (2000), Akeley K., (Ed.), Annual Conference Series, ACM SIGGRAPH / Addison Wesley Longman, pp. 385–392.

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-Frequency Shadows Using Non-linear Wavelet Lighting Approximation. *ACM Transactions on Graphics 22*, 3 (2003), 376–381.

[nVI] nVIDIA web page. http://www.nvidia.com.

[RSC87] REEVES W., SALESIN D., COOK R.: Rendering Antialiased Shadows with Depth Maps. In *Proceedings SIGGRAPH* (July 1987), pp. 283–291.

[Sil86] SILVERMAN B.: *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transactions on Graphics 21*, 3 (2002), 527–536.

[Sta94] STAM J.: Stochastic rendering of density fields. In *Graphics Interface '94* (May 1994), pp. 51–58.

[Wil78] WILLIAMS L.: Casting Curved Shadows on Curved Surfaces. In *Proceedings SIGGRAPH* (August 1978), pp. 270–274.

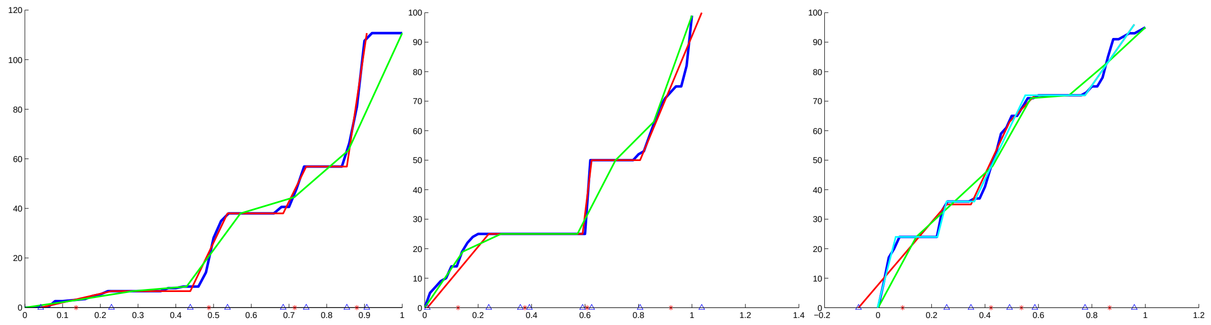

Figure 5: *Quality of reconstructed visibility function. The blue line represents the reference solution. The red and green line indicate the approximation using 4-means clustering and a uniform 8 bin histogram, respectively. The red stars and blue triangles on the horizontal axis indicate cluster means and bin extents, respectively. Left: results on data gathered from the hair model of figure 4. Middle and right: results for simulation with 4 and 8 gaussian distributions, respectively. The cyan line in the right figure shows the output of the clustering algorithm when scaling the initial clusters 75% towards the origin (cluster indications only correspond to solution in red).*

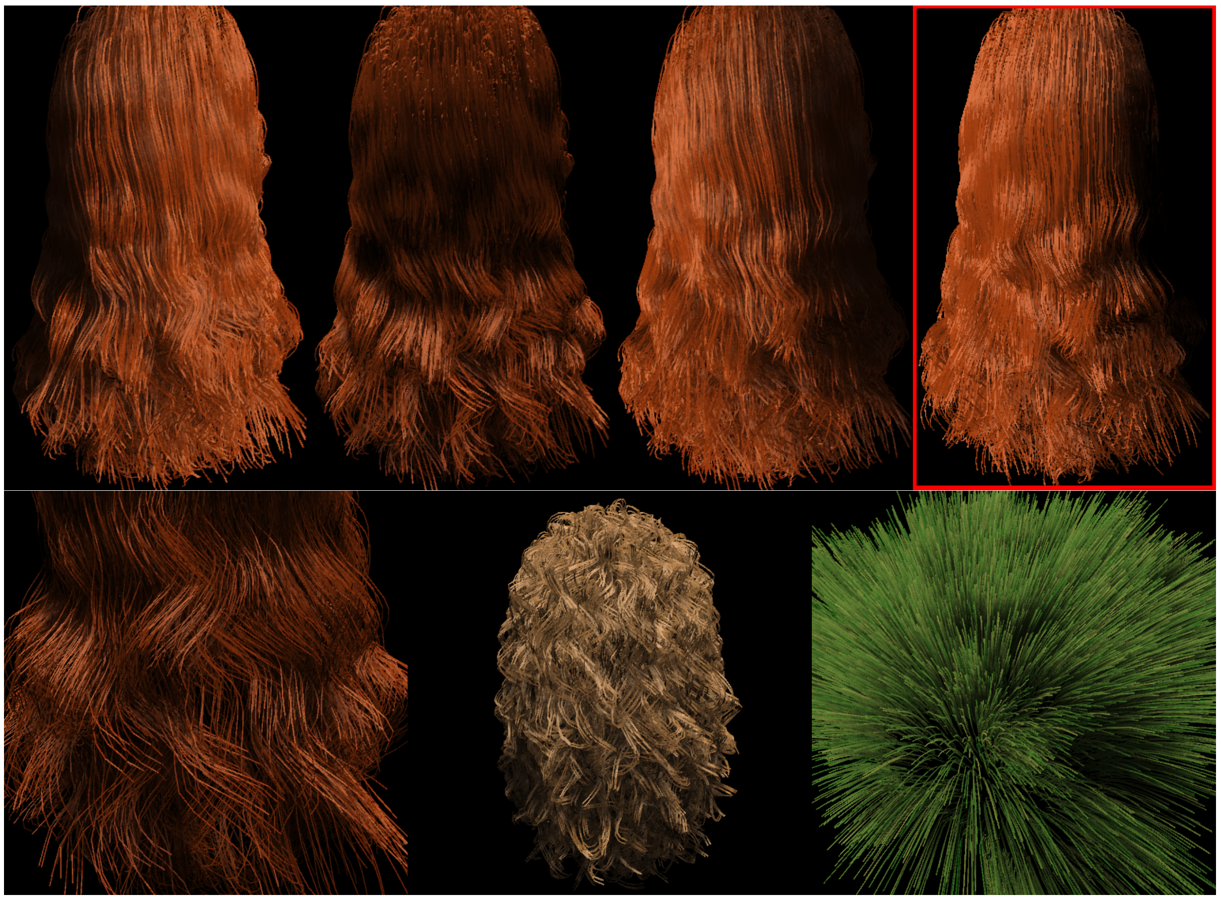

Figure 6: *Renderings of various models. Top row: red long hair consisting of 3600 strands (210K line segments), running at 3 Hz. The last image (with red border) show the result of running the opacity shadow map algorithm [KN01] using 256 slices, for the same setting as in the 3rd image. Bottom row, from left to right: closeup of the long hair; curly hair model, consisting of 3900 strands (200K segments); rendering grass as hair.*