Vision, Video and Graphics (2005)
E. Trucco, M. Chantler (Editors)

Realistic Real-Time Hair Simulation and Rendering

Yvonne Jung, Alexander Rettig, Oliver Klar, Timo Lehr

Fraunhofer IGD, Darmstadt, Germany

Abstract

We present a method for realistic rendering and simulation of human hair in real-time, which is suitable for the use in complex virtual reality applications. Neighbouring hairs are combined into wisps and animated with our cantilever beam based simulation system, which runs numerically stable and with interactive update rates. The rendering algorithm utilizes latest graphics hardware features and can even handle light coloured hair by including anisotropic reflection and internal transmission.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Animation; Color, shading, shadowing, and texture; Virtual reality

1. Introduction

The simulation of human hair is still an open area of research in computer graphics, for that reason alone, that a person usually has more than 100,000 hairs. The problem is getting by far more complex, if hair simulation is only one aspect of the dynamic behavior of virtual characters, which also have to show natural movements, gestures, face expressions, and speech within their virtual environment. Additionally besides the dynamic simulation of hair the realistic rendering of cloth, skin, eyes and certainly hairs is required (in Figure 1 some frames from our simulation are shown). Even with recent CPUs and programmable graphics boards many simplifications have to be made to be able to perform all these tasks together in real-time.

The demands of information rich dynamic virtual environments in general, like moving cameras, light sources and avatars, raise a lot of new problems with respect to both rendering and simulation: shaders have to be aware of moving light sources, back to front sorting depends on the camera position, collision detection has to deal with moving body parts. Thus for the use in real-time applications the large number of hairs and the many types of interdependencies like hair-body, hair-hair and hair-air interaction call for simplifying solutions.

Many simulation methods are based on mass spring damper systems, in which hair strands are modeled as line segments connecting some mass points, and the animation is done by iteratively computing the forces acting on them. The arising differential equations can be solved with explicit ODE solvers, which don't demand much computational power, but tend to blow off when forces become too strong. Implicit methods on the other hand like the backward Euler method are stable, but they are based on the computational expensive evaluation of big matrices, and constraints resulting from collisions make things even worse. Besides taking care of numerical stability there also is the problem of an adequate parameterization, otherwise the hair movement might resemble rubber bands or look like in an under water scenario.

Insofar we propose a wisp hair model based on quad strips, which is animated by a modified cantilever beam simulation. Rendering is done with GLSL (OpenGL Shading Language) shaders and includes amongst other things anisotropic reflection and a second specular highlight, which is characteristic for light colored hair.

Section 2 reviews previous approaches. Because we are focusing on simulation and rendering, the modeling aspect will only be covered shortly in part 3. In section 4 we will describe our cantilever beam based simulation system, and section 5 deals with rendering aspects like geometry sorting and lighting models. In chapter 6 some results are shown, which finally are discussed in section 7.

2. Related Work

In order to create convincing human hair there are basically four problems to solve: modeling and styling, hair dynam-

Figure 1: *Frames taken from an interactive hair simulation (body approximated with four spheres)*

ics, collision detection and response, and finally hair rendering [DMTKT93, NTSP02]. Presently a seamless transition between these categories is still problematic because the fewest hair simulation systems are self contained and they all differ in their geometrical representations, animation methods, and lighting models. Furthermore many approaches focus on special hair styles like fur or short hair [Gol97,GZ02].

Although the focus doesn't lie on modeling, a short review of common hair models is necessary, because not every hair model is suitable for every animation and rendering method. Thalmann et al. [NTSP02] classify hair models into several categories. The first one contains explicit models, which are relatively intuitive but computationally expensive, because every single hair strand is considered individually [AUK92, DMTKT93]. The next category are the cluster models, which utilize the fact that neighbouring hairs have similiar properties and tend to group together. They can be further divided up into hierarchical and flat models.

Ward et al. [WL03] propose a hierarchical level of detail representation, in which the hairs are either represented as precomputed strips, clusters or strands respectively. Simulation is done via a set of base skeletons and collision is handled with swept sphere volumes. More common are non-hierarchical schemes in which hair clusters are represented by generalized cylinders [KN02], trigonal prisms and polygon strips, which are rendered by using parametric surfaces in the method suggested by Koh and Huang [KH01].

Particle systems, especially the loosely connected particles method proposed in [BCN03], can be seen as an extension to clusters. Here the particles serve as sampling points for tracking hair dynamics in order to overcome the tight cluster structures during lateral motion. Volumetric textures are primarily used for very short hair [KK89]. The latest category regards hair as a fluid [HMT01]. Hair shape modeling is done by placing some ideal flow elements whereby collisions are implicitly avoided by the use of streamlines. In [Yu01] a method is suggested for adding curliness by modulating the hair strands with offset functions.

Animation can be done via key-framing or vertex perturbations on the one hand and numerical simulations on the other hand, reaching from mass spring systems over free form deformation and rigid multi-body chains up to vector fields [HMT01], based on the underlying hair model. A computationally cheap animation method also based on differential equations is the cantilever beam simulation originally proposed by Anjyo et al. [AUK92] for computing the hair bending of smooth hairstyles during the hair shape modeling process. The main drawback of most simulation methods is the fact, that depending on a hair style's complexity the simulators often can't guarantee for the hair style's preservation and for simplicity they mostly neglect the existence of head and body movement in collision detection.

But collision detection and response are a fundamental part of hair animation. Hair-hair collision for interactive applications is mostly ignored or quite coarsely approximated by bounding volume hierarchies, by density distributions [KLY02], by inserting some extra spring forces between neighbouring strips [KH01] or by generating auxiliary triangle strips between horizontally neighboured guide hairs against which collision is tested, and whose links can be broken or recovered dynamically based on their distance [CJY02]. Absolutely inevitable is the treatment of hair-head collision. Whilst geometry traversal, hierarchical or grid based schemes, and vector fields offer more precision, for real time applications a head can be approximated sufficiently with the help of spheres or ellipsoids [AUK92, KH01].

Last but not least rendering itself covers the full range from drawing single-coloured polylines [HR04] and alpha-textured polygonal patches [KH01] over heuristic local lighting models for anisotropic materials [Sch04] up to physically and physiologically correct illumination solutions [MJC*03]. Based on measurements of hair fibers the latter explains the existence of a second specular highlight visible by light hair by means of light paths described by a transmission- reflection- transmission- term *TRT* and simulates lighting with expensive ray tracing. In contrast the first two rendering methods are quite fast but not very convincing, because they don't take anisotropic reflection and self

shadowing into account, which are regarded as the most important lighting effects of hair by Koster et al. [KHS04]

An often referred reflectance model for dark hair, which exhibits higher reflectance than transmission and almost no self-shadowing, can be found in [KK89], and was extended for backlighting effects in [Gol97]. Based on Marschners et al.'s [MJC*03] results, Scheuermann [Sch04] recently extended this lighting model for the use in hardware shaders by perturbing the hair tangent for shifting both highlights.

Antialiasing of line drawing as a special case and alpha blending in general needs correct visibility ordering. In [Sch04] the blending problem is solved by drawing the appropriately pre-sorted hair patches in several passes. Kim and Neumann [KN02] suggest an ordering algorithm in which each line segment belongs to a slab perpendicular to the viewing vector subdividing the hair volume's bounding box in order to simplify the process of back to front rendering. In a similar manner opacity maps [KHS04], which discretely approximate a light intensity attenuation function for encoding a fragment's opacity value, are created from a light source's point of view, mostly in a preprocessing step.

In summary, all these models deal with certain aspects of hair modeling, simulation and rendering, but none of them is simply adoptable to our needs in that a stable, fast and convincing simulation method as well as realistic real-time shading for the use in complex virtual environments are taken into account.

3. Modeling

In order to reduce the geometric complexity and to avoid rendering problems we model hair wisps as relatively small quad strips instead of generating every single strand (see Figure 2): although polylines seem to be the natural representation of hairs at a first glance, they not only increase the load of the vertex processor, because a lot of lines are needed for a volumetric appearance, but also have to deal with aliasing artifacts due to the long thin nature of a hair which usually when projected is thinner than a pixel's size. Because of our fast simulation algorithm (see next section) we can model each hair strip from many small segments. Therefore we don't need smoother parametric representations like NURBS patches [KH01] and can take advantage of directly using the OpenGL support for rendering polygonal primitives.

The quad strips are appropriately layered on top of the scalp mesh for providing an impression of volumetric qualities. They can consist of a variable number of segments along the direction of hair growth, which is specified by the position of the hair root and an initial hair tangent vector. The hair distribution is defined by selecting some surface polygons on the scalp mesh. After having specified all other necessary parameters like width, height and number of segments as well as hair density the hair style is generated conferring to these parameters.

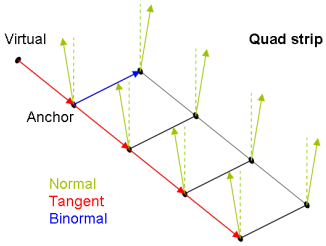

Figure 2: *Quad strip based hair model*

4. Dynamic Simulation

4.1. Structure

Our hair simulation is derived from the cantilever beam method [AUK92], which originally was intended for hair modeling only but not for its dynamics simulation. Compared to mass spring approaches, it provides a numerically simpler and - at least for smooth hair styles - visually more convincing way to simulate hair.

As already mentioned, mass spring systems are used for hair dynamics frequently. Mathematically these can be seen as damped oscillators which can be calculated with the help of differential equations by equating Newton's second law of motion ($F = ma = m\ddot{s}$) and Hook's law ($F = ks$), which relates the force F exerted by a spring with spring constant k and rest length l to its deflection $s = l' - l$.

Explicit numerical methods for solving these differential equations do not necessarily converge if forces are too strong and the size of the time step Δt lies above a certain threshold [BW98]. Whereas Anjyo et al. also used differential equations for computing the iteration steps during animation, our cantilever beam method works without them, which results in much higher simulation speed making it applicable for real time dynamics.

The most important difference of kinematic models like the cantilever beam model compared to an ordinary mass spring system is that the initial distance l between connected vertices can be fully conserved. Because neighbouring elements don't interact by means of spring forces, oscillations can't occur. Thus a kinematic simulation system keeps stable even with much bigger time steps.

Our modified cantilever beam algorithm internally works on a kinematic multibody chain, as illustrated in Figure 3. The nodes of the multibody chain are defined by the vertices of the original geometry and can be seen as joints connecting the edges between them. Two different types are distinguished, anchors and freemoving vertices. Anchors, resembling the hair roots, are connected to the scalp and serve as the attachment point of the chain, whereas all the other vertices in the chain are freemoving.

Freemoving vertices are moved only due to external

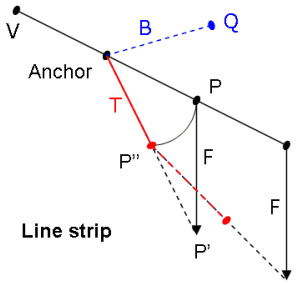

Figure 3: *Cantilever beam simulation*

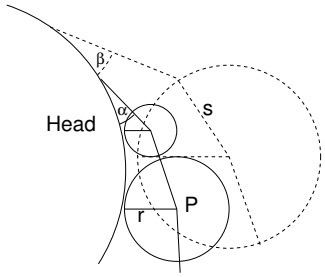

Figure 4: *Collision handling*

forces like gravity, the bending forces caused by their connected neighbour vertices and by applying the length conservation constraint. An external force F, e.g. gravity, which is acting on a chain link, results in a bending moment M, that causes a deflection of the actual segment along the direction of F. The calculation of the effect of this force is simplified by means of a heuristic approach: instead of calculating the torques, all forces, which are acting on the succeeding segments of a cantilever beam, are expressed by adding some offset vectors. Then length conservation is achieved by simply scaling the resulting vector back to the rest length l.

This method yields another advantage. During motion of the head, i.e. a coordinate transformation $\mathbf{T}$ of the head's frame of reference, it's not necessary to derive forces from $\mathbf{T}$ and feed them into the simulation to move the hair roots. It is sufficient to transform the hair root A directly accordingly to the new position ($A' = \mathbf{T} \cdot A$), because the transformation is then propagated through the chain intrinsicly keeping the hair length constant. While gaining a noticeable improvement in performance this way furthermore there are no stretching artifacts even when abrupt motions happen. Additionally to represent the inner tension of hair the angle between the bent hair segment and the direction of the previous hair segment is scaled down by a bend factor smaller than 1. This leads to very realistic looking hair motions.

For stabilizing the orientation of a hair wisp and aligning the beginning of the chain during movement according to the direction of hair growth a virtual vertex fixed to the scalp is introduced as predecessor of the anchor point (as shown in Figure 2). Together with the anchor it defines the growth direction during simulation. This way the hair strip's binormal B can be aligned horizontally to the head.

4.2. Collision Detection and Response

As has been stated earlier, besides a convincing simulation method a natural behavior in case of collisions is also required. Concerning hair, collision detection can be divided up into two types of interdependencies: hair-body and hair-hair interaction. Because of the large amount of hair the trade-off between quality and speed of the collision detection has to be taken into account.

Collisions with head or body are a hard constraint and must be treated explicitly. Tests have shown that because of the high self-occlusion of hair, users usually take no notice of a relatively low accuracy in collision detection between hair and head. Thus for approximation of the head we use parametric collision objects like spheres, ellipsoids and planes, for which intersection tests can be handled quite efficiently. If a freemoving vertex moves into a collision object, a penalty force is determined that projects the vertex back onto the surface.

Hair-hair collision can't be handled easily in real-time. Thus the interpenetration of hair wisps is avoided with a trick. On the one hand, hair strips are arranged on top of the scalp in different layers, each within a different distance to the head. For keeping this up during dynamics, each vertex P, depending on its position, is assigned a virtual collision sphere with a different radius r_P, in order to parameterize the distance to the head individually. On the other hand the problem is alleviated by using a slightly different bending factor for every chain, based on the position of its respective anchor. This layered collision avoidance structure is illustrated in Figure 4. As can be seen, hairs from lower levels aren't bent as much as those from higher levels. Besides this the vertices which are located nearer to the hair tip or which belong to hair wisps layered on top of the head are assigned bigger collision spheres than those from the bottom hair. This also has the nice side effect that implicitly collisions of hair segments s with the head are handled too, albeit with lower accuracy, although the algorithm explicitly only regards the vertices P.

4.3. Algorithm

As aforementioned the hair simulation is calculated on a skeleton, defined by one longitudinal edge of the quad strip. In order to re-transform the chain structure into a polygonal structure, every joint P_i has an associated point Q_i, which belongs to the second longitudinal edge. The vertices Q_i are calculated by adding the binormal vector B (the blue one in Figure 3) scaled by the initial strip width to the vertex P_i.

Normal, tangent and binormal are not only needed during rendering (see next section) but also for the simulation, which is done iteratively and consists of the following steps for the current time interval Δt:

1. For a simulation speed independent from the frame rate, scale the force offset vectors with an averaged time interval Δt_{avg}.

2. Transform all anchor points A, virtual points V and collision objects k to the new world coordinates.

3. $\forall$ anchors A calculate associated point Q, T_{avg} (difference vector between the last joint of a chain and A), tangent T, binormal B, and normal N:

$$T = A - V;\ B = T \times T_{avg};\ N = T \times B$$

4. $\forall$ joints P_i of every anchor A: (i) add offset vectors derived from forces; (ii) $\forall$ collision objects k with radius r_k: if P_i' is inside k then project P_i' back onto the surface at distance $r = r_k + r_{pi}$; (iii) add bending offset vector to P_i'; (iv) calculate vectors T_i and N_i:

$$T_i = P_i' - P_{i-1}';\ N_i = T_i \times B$$

(v) keep initial segment length l_i:

$$P_i'' = P_i' + T_i(\frac{l_i}{|T_i|} - 1)$$

5. Rendering

5.1. Sorting

In order to avoid aliasing and to overcome the rectangular structure during rendering, alpha blending, with disabled depth buffer write, is used in combination with usual RGBA textures, which are mapped onto the hair patches. In this way, an impression of thin, semi-transparent hair is created. Alpha blending only works if the hair geometry is rendered last and sorted back to front along the viewing vector, otherwise the edges of the underlying geometry, like the head, would be visible. Because neither the viewpoint nor the virtual character can be assumed to be immobile, sorting has to be done dynamically.

It is obvious that a naive per quad sorting must fail, because the hair patches are very densely neighboured. Besides this, the algorithm can not always guarantee that there is no self pervasion. Insofar no unique sorting sequence can be given, which takes the connection of the segments of a hair patch into consideration. A simple solution to this problem is to sort the primitives at the next level of hierarchy.

Back to front sorting of the quad strips solves the latter problem but arises another artifact, caused by the relatively large extension of the surface compared to one single reference point chosen for sorting purposes. Then the roots of wisps, which are lying nearer to the camera, seem to be layered on top of hair wisps being further away, although in reality the hair position is view-independent. This can be alleviated by sorting the uppermost strips along the head's up-vector in a second step. To our experience an empirically determined factor of about fifteen percent of all non-occluded hair strips produces quite satisfactory results.

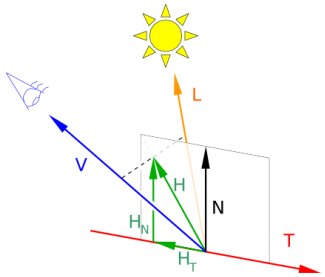

Figure 5: *Tangent based lighting model*

5.2. Lighting Model and Shading

5.2.1. Reflection Properties of Human Hair

The long thin nature of hairs, which usually are aligned in one predominant direction and microscopically consist of steplike seceding segments, can be regarded as the microstructure of a hair style. It therefore contributes to the hair's anisotropic reflection properties, because the normal distribution along the hair fibres is different from the distribution across them.

The first impressive results for hair rendering already have been achieved by Kajiya and Kay [KK89], who also suggested a very common phenomenologically motivated local lighting model, which still constitutes the basis for many modern rendering approaches concerning hairs. The demand for photorealism however calls for a shift from phenomenologically based lighting models to physically correct lighting simulations.

Despite the great advances in the field of graphics hardware, there still remains a trade-off between visual quality and rendering speed. Because anisotropic reflection can not be evaluated with standard OpenGL, nowadays rendering is often directly done on the GPU by using hardware shaders, with which nearly photo-realistic realtime effects can be reached.

Hair fibers can be regarded as cylinders which are nearly infinitesimal small in diameter. Unlike it's the case with surfaces for a given point P on the hair fibre there exists an infinite number of normals lying in a plane orthogonal to the fiber's tangent T. A normal N suitable for lighting calculations is the normal which is coplanar with the half vector $H = \frac{L+V}{|L+V|}$ and the tangent T (see Figure 5).

This normal vector doesn't need to be computed explicitly for calculating the intensity of the specular highlight, de-

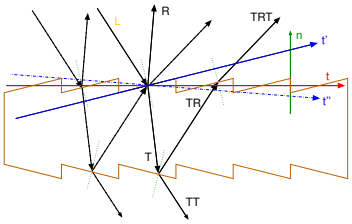

Figure 6: *Transmission and Reflection*

scribed by $(H \bullet N)^s$ in the Blinn-Phong illumination model, when making use of an orthogonal decomposition of the half vector $H = H_N + H_T$. Then the specular term can be calculated solely in terms of H and T (both being of unit length) as follows:

$$H \bullet N = |H_N| = \sqrt{1 - |H_T|^2} = \sqrt{1 - (H \bullet T)^2}$$

5.2.2. Specular Highlights

The rendering of long, light colored hair is by far much more complex than of short, dark hair. Thus due to the translucency characteristics of hair fibres the additional consideration of transmission, dispersion and self shadowing is required. As described in Marschner et al. [MJC*03], in case of direct lighting, there are two different specular highlights. The first highlight results from direct reflection R at the surface of a hair fibre. Caused by the thin tilted squamous structure it steps somewhat shifted up along the tangent towards the hair root.

The second highlight TRT, which does not arise with black hair, results from internal reflection (Figure 6). The incident light passes through the interior of the fibre and is reflected at the opposite side of the cylindric shape. Because of refraction the light's direction changes when passing through two media with different densities, so the secondary peak appears enervated and shifted towards the hair tip. The highlight is more like a glint and because of its way through the medium it gets colored by the pigments of the hair. Likewise only with lighter hair the transmission-transmission term TT produces backlighting effects.

A physically correct treatment exceeds the capacities of a real time application. Nevertheless, in order to calculate the different peaks described above (second highlight shown exaggerated in green in Figure 7 on the left), after Scheuermann [Sch04], two tangents T' and T'' are needed, which are shifted in opposite directions. This can be achieved by adding a scaled normal onto the original tangent T (diagrammed in Figure 6). The observed dispersion of light, caused by scattering in the interior of the medium, is simulated by a noise function which for efficency reasons is stored in a texture and can easily be accessed by a simple texture look-up in the fragmentshader.

To achieve best results, the rendering equation is calcu-

Figure 7: *Highlights and Shadows*

```
1  vec4 hairTexCol = texture2D(hairTex, gl_TexCoord[0].st);
2  if (hairTexCol.a <= 0.1) discard;
3  else {
4  float shiftTex = texture2D(noise, gl_TexCoord[0].st);
5  float wrapDiffuse = max(0, (dot(vL, vN)
       + scattering) / (1 + scattering));
6  vec3 diffuse = wrapDiffuse * diffuseColor;
7  vec3 ambient = ambientColor * vC;
8  vec3 specular1 = specularColor1 * calcHighlight(
       vT, vN, vH, shininess1, shiftValue1 + shiftTex);
9  vec3 specular2 = specularColor2 *
       shiftTex * calcHighlight(vT, vN, vH,
       shininess2, shiftValue2 + shiftTex);
10 vec3 color = (specular1 + specular2 +
       diffuse + ambient) * hairTexCol.rgb;
11 gl_FragColor = vec4(color, hairTexCol.a);
12 }
```

Algorithm 1: *Fragmentshader code snippet*

lated per fragment on the GPU. The vertex shader computes all necessary vectors and passes them to the fragment unit. In Algorithm 1, variables with the prefix *v* indicate (GLSL specific) varying parameters, which are passed over from the vertex to the fragment processor. The pixel shader calculates the diffuse and specular term for the lighting model and another term for describing the ambient light that has been scattered around for several times.

With help of the fragment processor's *discard* command (line 2 of Algorithm 1), fragments which satisfy the condition $\alpha < \varepsilon$ with a certain $\varepsilon > 0$ are thrown away. Defining areas in the alpha map which shall be ignored in further lighting calculations not only saves computation time but also has another advantage. With simple alpha blending and no special shadow pass materials, shadows which are cast from the hair patches onto the character's head would be shaped like a whole patch instead of single hairs, because transparency is disregarded in depth mapping. By discarding certain pixels, good shadowing results are achieved even with simple depth maps (Figure 7, right).

5.2.3. Ambient and Diffuse Lighting

Scattered light, which leaves the backfacing side of a hair fibre (see TT in Figure 6), is simulated by a scattering function on the GPU by extending the diffuse Lambert term $N \bullet L$. The trick with the so called 'wrap lighting' is to 'turn' the light around the object. Therefore a term $w \in [0;1]$ is added to the diffuse term (line 5 of Algorithm 1), which causes a certain light intensity on the backfacing side (see Fernando et al. [Fer04], p. 264).

To improve the impression of a hair volume, the normals

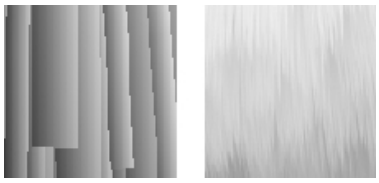

Figure 8: *Normal Bending and Ambient Occlusion*

of each patch are bended into the direction of the binormals (see Figure 2, and Figure 8, left). Thus the light intensity is sloping towards the edges, which leads to the impression of a cylindrical shape.

A technique for approximating complex light distributions is called 'ambient occlusion' [Fer04]. The basic idea is to calculate for each point to be lit the fraction of incident direct light that is not occluded by any other geometry. This self-shadowing information will be used to scale the lighting term. The original method consists of several rendering passes, nevertheless it can be simplified with some object knowledge, thus hair from lower layers receive less energy than those from top layers. This occlusion term is calculated per vertex depending on its position and the height of its associated anchor point (see line 7 of Algorithm 1, and right side of Figure 8).

6. Results

In contrast to the often used mass spring systems our heuristically motivated hair simulation system is computationally less intensive because of the special chainlike structure of the hair wisps. Especially properties like pliancy can be easily included without any additional expense. Moreover the system is nearly non-oscillating and insofar it is most unlikely that it will blow off. Therefore the number of iterations per time-step can be reduced a lot in favour of a much higher frame-rate. Concerning the lighting simulation, we are likewise still working on a phenomenological level by adopting approaches from physically based rendering to simpler GPU-based methods. However, the result is very close to photorealism and is applicable in information rich virtual environments. As can be seen in Figures 9 and 10, a realistic appearance with interactive frame rates is achieved even for light colored hair.

The simulation and rendering components were implemented in C++ as native scene graph nodes in our inhouse VR/AR system Avalon, which supports an extension of X3D as the application description language. This is done by defining a simulation system scene graph node, and another scene graph node for rendering a set of sorted primitives, whose field values are updated by the simulation system node via the X3D routing mechanism. Therefore the simulation system as well as the hair shaders are easily to create, use and parameterise even during runtime.

# Vertices	5096	18690	37498	52192
# Anchors	364	623	2316	2388
Simulation	248 fps	80 fps	37 fps	18 fps
Rendering	64 fps	49 fps	19 fps	16 fps
Combined	52 fps	35 fps	15 fps	11 fps

Table 1: *Benchmarks for different hair styles*

Figure 9: *Blond hair viewed from behind*

Benchmarks are shown in Table 1 for four different numbers of vertices resp. anchor points. The values have been taken using a Linux PC with Pentium IV, 2.8 GHz with Hyperthreading, 1 GB RAM with a GeForce FX 5900 Ultra graphics card (about 170 000 pixels are covered by hair). The first row shows animation performance with standard OpenGL lighting, the second shows performance with GLSL shaders but without simulation, and the last row shows frame rates achieved with both simulation and shaders activated.

Figure 10: *Grey hair (circa 37,000 vertices)*

7. Conclusions and Future Work

In this paper we analyzed existing approaches for hair simulation, whereas most of them are only suitable for offline rendering. We then proposed a realistic looking method for simulating and rendering human hair in real time, applicable in complex scenarios. By including both transmission based lighting terms *TT* and *TRT*, an almost photo-realistic appearance even for light colored hair is achieved. Moreover on modern graphics hardware our hair shader is fast enough for the use in real time applications, at least if the hairs don't cover the whole screen. Because of the chainlike structure and some simplifications, our cantilever beam based simulation system is fast, looks convincing and runs numerically very stable.

The main drawback is the problem, that without any further extensions the simulator is only suitable for smooth hair styles. Because of this, we are investigating the possibility of extending our system for the simulation of other hair styles. Currently we are working on an adaptive hierarchical simulation algorithm, which emphasizes the dynamic transition from clustered hair wisps to single hair strands in consideration of external factors like viewing distance or speed of movement. Furthermore we are implementing a texture based modeling tool for simplifying the process of creating different hair styles.

References

[AUK92] ANJYO K.-I., USAMI Y., KURIHARA T.: A simple method for extracting the natural beauty of hair. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (1992), ACM Press, pp. 111–120.

[BCN03] BANDO Y., CHEN B.-Y., NISHITA T.: Animating hair with loosely connected particles. *Computer Graphics Forum 22*, 3 (9 2003), 411 – 418.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. *Computer Graphics 32* (1998), 43–54.

[CJY02] CHANG J. T., JIN J., YU Y.: A practical model for hair mutual interactions. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), ACM Press, pp. 73–80.

[DMTKT93] DALDEGAN A., MAGNENAT-THALMANN N., KURIHARA T., THALMANN D.: An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum 12*, 3 (1993), 211 – 221.

[Fer04] FERNANDO R. (Ed.): *GPU Gems*. Addison Wesley, 2004.

[Gol97] GOLDMAN D. B.: Fake fur rendering. In *SIGGRAPH '97: Proc. of the 24th annual conference on Computer graphics a. interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 127–134.

[GZ02] GUANG Y., ZHIYONG H.: A method of human short hair modeling and real time animation. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (2002), IEEE Computer Society, p. 435.

[HMT01] HADAP S., MAGNENAT-THALMANN N.: Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3 (September 2001).

[HR04] HERNANDEZ B., RUDOMIN I.: Hair paint. In *CGI '04: Proceedings of the Computer Graphics International* (2004), IEEE Computer Society, pp. 578–581.

[KH01] KOH C. K., HUANG Z.: A simple physics model to animate human hair modeled in 2d strips in real time. In *Proceedings of the Eurographic workshop on Computer animation and simulation* (2001), Springer-Verlag New York, Inc., pp. 127–138.

[KHS04] KOSTER M., HABER J., SEIDEL H.-P.: Real-time rendering of human hair using programmable graphics hardware. In *CGI '04: Proceedings of the Computer Graphics International* (2004), IEEE Computer Society, pp. 248–256.

[KK89] KAJIYA J. T., KAY T. L.: Rendering fur with three dimensional textures. In *SIGGRAPH '89: Proc. of the 16th annual conference on Computer graphics a. interactive techniques* (1989), ACM Press, pp. 271–280.

[KLY02] KONG D., LAO W., YIN B.: An improved algorithm for hairstyle dynamics. In *ICMI '02: Proceedings of the 4th IEEE International Conference on Multimodal Interfaces* (2002), IEEE Computer Society, p. 535.

[KN02] KIM T.-Y., NEUMANN U.: Interactive multiresolution hair modeling and editing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 620–629.

[MJC*03] MARSCHNER S. R., JENSEN H. W., CAMMARANO M., WORLEY S., HANRAHAN P.: Light scattering from human hair fibers. *ACM Trans. Graph. 22*, 3 (2003), 780–791.

[NTSP02] N.MAGNENAT-THALMANN, S.HADAP, P.KALRA: State of the art in hair simulation. In *International Workshop on Human Modeling and Animation, Korea Computer Graphics Society* (2002), pp. 3–9.

[Sch04] SCHEUERMANN T.: Practical real-time hair rendering and shading. Siggraph04 Sketches, August 2004.

[WL03] WARD K., LIN M. C.: Adaptive grouping and subdivision for simulating hair dynamics. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003), IEEE Computer Society, pp. 234 – 242.

[Yu01] YU Y.: Modeling realistic virtual hairstyles. In *Proceedings of Pacific Graphics* (2001), pp. 295–304.