Real-time Hair Simulation on GPU with a Dynamic Wisp Model

Masaki Oshita
Kyushu Institute of Technology
680-4 Kawazu, Iizuka City, Fukuoka, Japan.
oshita@ces.kyutech.ac.jp

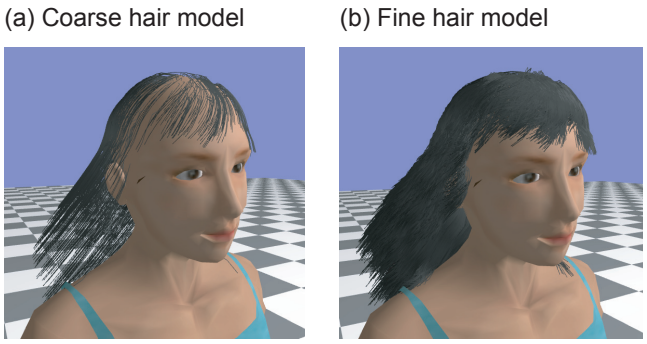

Figure 1: Overview of the proposed method. (a) The movement of a coarse hair model is simulated using a particle-based simulation. (b) A fine hair mode is then generated geometrically using a dynamic wisp model.

ABSTRACT

In this paper, we present a method for real-time hair animation. We combine a conventional particle-based dynamic simulation and a dynamic hair generation technique. First, the movements of a small number of hairs (coarse model) are simulated using a dynamic simulation. Since this stage uses only a small number of hairs, the simulation is quick. A larger number of hairs (fine model) are then generated from the coarse model using a dynamic wisp model. The shape of a wisp and the shapes of the individual strands are geometrically controlled based on the velocity of the corresponding particle in the coarse model. This model simulates hair-hair interactions between the strands in the hair wisps. Our method is designed to work on a GPU, and generates a realistic hair animation in real-time.

Keywords: hair simulation, GPU, wisp model, geometric deformation.

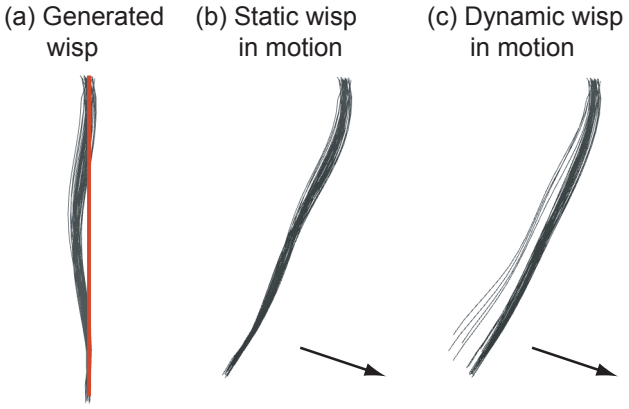

Figure 2: Dynamic wisp model. (a) Wisp strands are generated around a simulated strand. (b) Static wisp in motion. (c) Dynamic wisp in motion. The shape of wisp and the shape of individual strands are controlled based on the velocities of the simulated hair strand.

1. INTRODUCTION

Hair simulation is one of the important techniques in computer animation. Researchers have been developing various methods for modelling, simulating, and rendering hairs. Currently most methods are for off-line animations. Since human hair consists of more than 100,000 hair strands, dealing with so much data is a very time-consuming process. Real-time hair simulation is needed for online animations such as computer games, virtual reality and so on. However, it is still a challenge.

The most popular technique for hair simulation is a particle-based method. By modelling a hair as a serial link of small particles, the movement of hairs can be simulated based on the forces that act on all particles. However, even with the latest computers and techniques, simulating a whole hair of a person takes a few seconds for each frame of the animation. By reducing the number of hairs and simplifying the simulation model, real-time simulation can be achieved. However, such simulation lacks reality. Currently very few computer games use such a simplified hair simulation. Instead, many of them still use polygon-based motionless hair.

In this paper, we propose a real-time hair simulation method,

which combines a particle-based hair simulation with a dynamic wisp generation model (Figure 1). We use a small number of hair strands for the dynamic simulation (coarse model). Each simulated hair strand is then considered as a hair wisp and about ten to twenty hair strands are generated for each hair wisp (fine model). Since we only use between a few hundred and a thousand hairs in the dynamic simulation, it works very efficiently. This coarse-to-fine approach has been used in many other methods [6][5][14][3]. The main contribution of our work is that we introduce the dynamic wisp model. The shape of a wisp and the shape of individual strands are controlled based on the velocities of hair particles (Figure 2). The shape of a wisp gets wider in response to the horizontal velocity. The shape of individual strands also shrinks horizontally due to horizontal pressure.

Simulating the interactions between individual hairs is very important in hair simulation. However, it has been omitted in many online simulation systems. As a result, the hairs tend to move uniformly. Our method models the interactions of hair strands in hair wisps. Although our dynamic wisp model is based on a geometric deformation and not a physically correct model, it is able to simulate the nature of the dynamics of hair wisps efficiently.

Our method is also designed to work on a GPU. Using graphics hardware (GPU) for dynamic simulation has received a great deal of attention recently. Since each particle can be handled in parallel, particle-based simulations can easily be implemented on a GPU [14][15][18]. However, handling hair-hair interactions is very difficult, since it needs to take into account multiple particles at any one time. Our method deals with such hair-hair interactions using the dynamic wisp model and without handling multiple particles. In addition to using a similar method to that in [14] for the simulation of the coarse model on a GPU, our dynamic wisp model is also implemented on the GPU. As a result, our method works in realtime. We give a detailed explanation of our implementation in this paper.

The rest of this paper is organized as follows. In Section 2, we review related work. Sections 3 and 4 give an overview of our system and particle systems, respectively. In Section 5, the dynamic wisp model is explained. Section 6 gives some experimental results together with a discussion. Finally, we present our conclusions in Section 7.

2. RELATED WORK

Much research has been conducted on hair simulation. In this section, we would like to focus on methods for handling hair-hair interactions and methods for generating fine hair models from coarse ones.

2.1 Hair-hair Interactions

Some unique models have been proposed by researchers to handle hair-hair interactions efficiently. Bando et al. [1] proposed a loosely connected particle system. They use separate particles to model the whole hair and render it using point textures. Although their method simulates the movements of the whole hair efficiently, they cannot handle various hairstyles and realistic hair wisps and strands. Hadap et al. [9] considered hairs as a continuum and introduced a fluid simulation to simulate the movement of hair volumes. The individual hair strands are then simulated so that they follow the fluid. This method still requires a great deal of computational time as both the fluid and the individual hairs need to be simulated, although the interactions between individual strands need not be handled. Volino et al. [17][8] used a grid model to simulate the movement of hair volumes. They used free-form deformation to simulate the deformation of the grid and individual hairs are then geometrically deformed based on the grid. This method can handle complex hair styles efficiently without computing the interactions between individual hair wisps or strands. However, the shapes of hair wisps and individual hairs are consistent and it lacks reality. Our method is also based on a geometric deformation, but it simulates natural hair wisps rather than the whole volume of hair. The earlier methods tried to introduce novel models to simulate the movement of hair volumes including the effects of hair-hair interactions without actually computing the interaction between individual hairs. Our method focuses on the effects of hair-hair interactions on hair wisps. We think that the movement of hair volumes can be simulated with existing particle-based systems and that dynamic hair wisps are more important.

2.2 Dynamic Generation of Fine Hair Model

Methods exist for generating a fine hair model from a coarse hair model. Using a coarse hair model for modelling and simulating hairs is a reasonable approach. However, in order to generate realistic images, we still need a fine model for rendering.

Interpolating nearby hairs [5][14] is a commonly used method. However, real human hair strands tend to stick to each other and to group themselves into wisps rather than individually. Therefore, the simple interpolation is not the best method.

Wisp or cluster models are also a popular method. Watanabe et al. [22] proposed a wisp model in which a hair wisp consists of a number of hair strands with direction and length variation. Yang et al. [19] introduced a cluster model in which a hair wisp (cluster) is modelled as a generalized cylinder and is rendered with a randomly generated density volume map. In these models, the shape of wisps and strands is not completely controllable. Choe et al. [6] used a statistical model to generate the wisp shapes. The individual hairs in a wisp are generated based on given parameters such as a length distribution function, radius function, and fuzziness. They also used a statistical model to generate waving hair strands by combining small segments from given sample hair strands. Their system is aimed at editing static hair styles and dynamic movements are not considered in their model. Yu [20] and Kim et al. [11] also developed wisp-based hair style modelling systems. Ward et al. [21] introduced level-of-detail representation of hair wisps. Our method uses similar parameters to those in [6][20] to generate hair wisps. However, instead of using constant parameters, we control the parameters based on the velocity of particles and change the shape of hair wisps and individual hair strands dynamically.

3. SYSTEM OVERVIEW

An overview of our system is shown in Figure 3. Our method works on a GPU and all variables are stored as textures on the video memory of the graphics hardware. The sys-

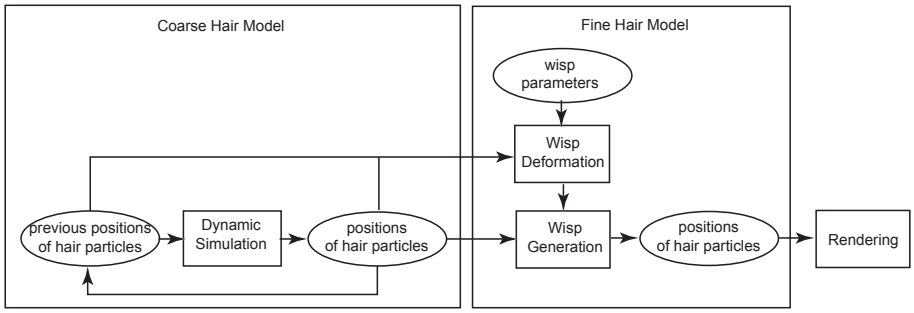

Figure 3: System overview.

tem consists of two stages; dynamic simulation of a coarse hair model and generation of a fine hair model. First, the positions of the particles of the coarse hair model and the wisp parameters for generating the fine hair model are initialized. In the dynamic simulation, the movements of the particles are computed using a particle-based physics simulation based on the movement of the human body. A fine hair model is then generated based on the simulated coarse hair model and the given parameters. A hair wisp (a number of strands) is generated around each simulated hair strand. The shapes of the hair wisps and strands are also controlled based on the velocities of the corresponding particles in the coarse model which are computed from the previous and current particle positions. Finally, the generated wisps are rendered on the screen.

4. DYNAMIC SIMULATION

As explained in Sections 1 and 3, we use a conventional particle-based method [1][14][5] for dynamic simulation of the coarse hair models. In this section, we briefly describe our model and its implementation on a GPU.

4.1 Particle Model

Each hair strand is modelled as a serial link of particles. In our implementation, ten to twenty particles are used for each strand, depending on its length. The movements of particles are computed using numerical integrations based on the forces acting on particles, such as spring forces between adjacent particles to keep the hair length constant, collision forces from the human body and from nearby hairs, gravitational force and so on.

4.2 Integration

We use Verlet integration to compute the movements of particles [14][15], because it is more stable than Euler integration with large time steps and does not need to store particle velocities explicitly. The position of the k-th particle $\mathbf{P}_k$ is updated based on the previous two frames' positions $\mathbf{P}_k^{-1}$ and $\mathbf{P}_k^{-2}$, gravity acceleration $\mathbf{g}$, and wind force $\mathbf{w}$ as follows,

$$\mathbf{P}_k = \left(\mathbf{P}_k^{-1} - \mathbf{P}_k^{-2}\right)\Delta t + \mathbf{g}\Delta t^2 + \frac{|\mathbf{w} \times \mathbf{t}_k|}{|\mathbf{w}|}\mathbf{w}\Delta t^2, \qquad (1)$$

where $\mathbf{w}$ is a given wind force which has a direction and magnitude. We assume that wind acceleration to each particle is proportional to the angle between the hair direction (tangent vector) $\mathbf{t}_k$ at the particle and the direction of the wind force $\mathbf{w}$.

4.3 Constraints

Spatial constraints are introduced to keep the hair length constant and to avoid penetration into the human body. In this step, the particle positions are directly modified to satisfy constraints in a similar way to [13][15][18]. The distance from the next particle $\mathbf{P}_{k-1}$ is kept constant as follows,

$$\mathbf{P}'_k = \mathbf{P}_k + \frac{l_k - |\mathbf{P}_k - \mathbf{P}_{k-1}|}{|\mathbf{P}_k - \mathbf{P}_{k-1}|}\left(\mathbf{P}_k - \mathbf{P}_{k-1}\right), \qquad (2)$$

where $l_k$is the initial length of the link between $\mathbf{P}_k$ and $\mathbf{P}_{k-1}$. The hair-body constraints are also applied in the same way. The body is represented as a small number of spheres which have a center position $\mathbf{o}_i$ and radius r_i.

$$\mathbf{P}'_k = \mathbf{P}_k + \frac{r_i - |\mathbf{P}_k - \mathbf{o}_i|}{|\mathbf{P}_k - \mathbf{o}_i|}\left(\mathbf{P}_k - \mathbf{o}_i\right). \qquad (3)$$

The positions of the spheres are updated based on the posture of the human figure in each frame.

The position of the base particle of each strand is constrained by the movement of the head. It is updated before applying other constraints based on the current position and orientation of the head.

As explained before, the hair-hair interactions are not handled at this stage, as it is difficult to do so efficiently. In contrast to cloth simulations [15][18] in which many particles are connected to each other, hair particles are connected to only two adjacent particles, which makes the implementation much simpler and faster.

4.4 Implementation on GPU

The particle-based simulation can be processed on a GPU. The particle positions are stored as colors on a texture image as shown in Figure 4. We need at least three textures for $\mathbf{P}$, $\mathbf{P}^{-1}$ and $\mathbf{P}^{-2}$. An array of texture coordinates, in which the particle positions are stored, is given to the fragment shader programs.

For integration, a shader program computes the output position $\mathbf{P}$ by using the previous positions $\mathbf{P}^{-1}$ and $\mathbf{P}^{-2}$. To simplify the shader program, all particles of a strand are recorded on the same line of the texture so that the position of the adjacent particle can be accessed easily. This is necessary when the shader program computes a tangent vector $\mathbf{t}_k$ from the adjacent particle's position. In addition, we add a dummy particle before the base particle of the strand in

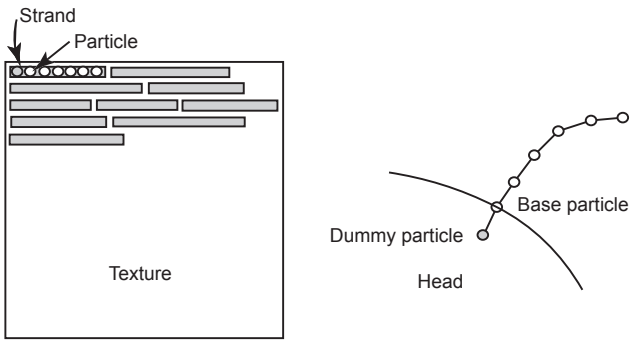

Figure 4: Particle positions on a texture.

order to compute $\mathbf{t}_0$ (Figure 4). The position of the dummy particle is computed based on the movement of the head as well as the base particle. The initial distance l_k can be stored as an alpha value of $\mathbf{P}$ when we use RGBA texture. Alternatively, we can use a luminance (1D) texture for l_k.

For applying constraints, shader programs compute the output positions $\mathbf{P}'$ by using the integrated positions $\mathbf{P}$. Since we do not need $\mathbf{P}^{-2}$ anymore, we can use the texture for $\mathbf{P}^{-2}$ as the texture for $\mathbf{P}'$. The constraints should be applied iteratively. Therefore, the length and collision constraints are implemented as different shader programs and they are applied interactively a few times.

5. DYNAMIC WISP MODEL

Hair wisps are generated for each individual hair strand that is simulated in the previous stage. In this paper, we refer to the hair strands used in the dynamic simulation as 'Master Strands', and the hair wisp strands generated around a master strand as 'Member Strands', which is consistent with [6]. The positions of the particles of member strands are computed from a master strand using a parametric wisp model. In our simulation, about ten to twenty member strands are generated from a master strand.

In this section, we first explain a static parametric hair wisp model (Section 5.1) and then extend it to a dynamic model (Section 5.2).

5.1 Static Wisp Model

We use similar parameters to those in [6][20][11]. We have chosen a small number of parameters for our wisp model, namely the shape of hair wisps (distribution and distances between individual hair strands) and the shape of hair strands (relative positions of individual particles to a straight hair strand).

5.1.1 Shape of Wisps

For the shape of a hair wisp, we use a length distribution function $D(l)$ and radius function $R(s)$, as shown in Figure 5, in a similar way to [6]. $D(l)$ represents the probability of a master strand whose length is $l\,(l>0)$. $R(s)$ represents the radius of the wisp at the point $s\,(0 \le s \le 1)$ where $s=0$ is the root of the strand and $s=1$ is the top of the strand. $D(l)$ and $R(s)$ are given by the user for individual master strands. Based on$R(s)$, the horizontal offsets (relative

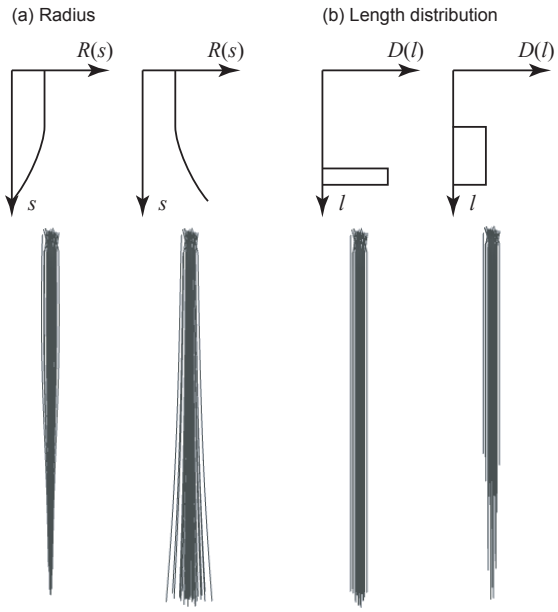

Figure 5: Wisp shape parameters. (a) Radius function, and (b) length distribution function.

positions to the particles of the master strand) of member strands are initialized as follows,

$$\begin{aligned} \alpha &= rand\,(0.0 \sim 1.0) \\ \beta &= rand\,(0.0 \sim 2\pi) \\ u_{i,j,0} &= R\,(0)\,\alpha \sin(\beta) \\ v_{i,j,0} &= R\,(0)\,\alpha \cos(\beta)\,, \end{aligned} \tag{4}$$

where $u_{i,j,0}, v_{i,j,0}$ is the horizontal offset of the base particle of the j-th member strand of the i-th master strand.

Based on $D(l)$, the length of the j-th member strand of the i-th master strand $l_{i,j}$is determined by,

$$\int_0^{l_{i,j}} D\,(l)\,dl = rand\,(0.0 \sim 1.0)\,. \tag{5}$$

The offsets of the following particles are computed using the radius function,

$$\begin{aligned} u_{i,j,k} &= \tfrac{R(k/m)}{R(0)} u_{i,j,0} \\ v_{i,j,k} &= \tfrac{R(k/m)}{R(0)} v_{i,j,0} \\ t_{i,j,k} &= k\,(l_{i,j} - l_i)\,/m. \end{aligned} \tag{6}$$

The offset of the k-th particle of the j-th member strand of the i-th master strand $\mathbf{q}_{i,j,k} = (u_{i,j,k}, v_{i,j,k}, t_{i,j,k})$ is represented in the local coordinates on the particle of the master strand $\mathbf{p}_{i,k}$ (Figure 6). The local coordinates $(\mathbf{t}_{i,k}, \mathbf{u}_{i,k}, \mathbf{v}_{i,k})$are computed based on the tangent vector $\mathbf{t}_{i,k}$ and a reference vector $\mathbf{x}$ (We use the x-axis of the head as the reference vector. We do not use y-axis as the reference vector because the hair strands are often vertical).

$$\mathbf{u}_{i,k} = \mathbf{x} \times \mathbf{t}_{i,k}, \quad \mathbf{v}_{i,k} = \mathbf{t}_{i,k} \times \mathbf{u}_{i,k}. \tag{7}$$

The global position of particle $\mathbf{p}_{i,j,k}$ is computed from the offset $\mathbf{q}_{i,j,k} = (u_{i,j,k}, v_{i,j,k}, t_{i,j,k})$ and the local coordinates

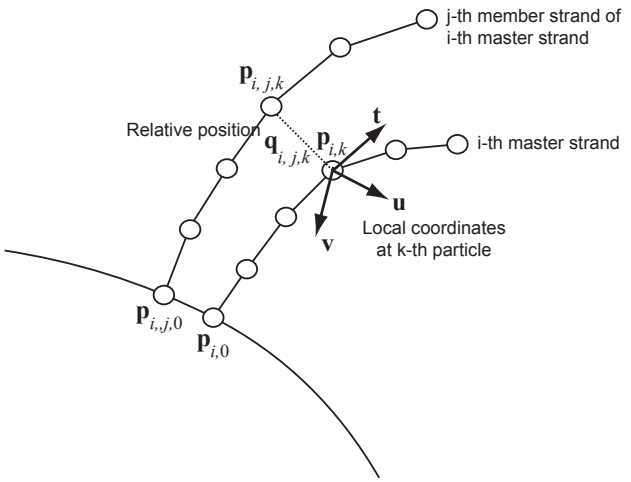

Figure 6: Spatial relationship between master and member strands. A particle position of a member strand is represented in the local coordinates of the corresponding particle of the master strand.

$(\mathbf{t}_{i,k}, \mathbf{u}_{i,k}, \mathbf{v}_{i,k})$,

$$\mathbf{p}_{i,j,k} = \mathbf{p}_{i,k} + t_{i,j,k}\mathbf{t}_{i,k} + u_{i,j,k}\mathbf{u}_{i,k} + v_{i,j,k}\mathbf{v}_{i,k}. \quad (8)$$

5.1.2 *Shape of Strands*

The straight base lines of the member strands are computed using the above model. In addition, curved member strands are generated based on the straight line and curve parameters. According to [20], most hair strands can be modelled using a sinusoidal curve and a radius function. Since we already use a radius function for the shape of wisps, we use only a sinusoidal curve here,

$$\begin{array}{l} cu_{i,j,k} = \alpha_u \sin(fs + \theta) \\ cv_{i,j,k} = \alpha_v \cos(fs + \theta) \end{array}, \quad (9)$$

where $(cu_{i,j,k}, cv_{i,j,k})$represents an offset for the strand shape and $(\alpha_u, \alpha_v, f, \theta)$ are curve parameters that represent the two radii, frequency of wave, and phase of wave, respectively. The offset for the strand shape is added to the offset for the wisp shape.

$$\begin{array}{l} u'_{i,j,k} = u_{i,j,k} + cu_{i,j,k} \\ v'_{i,j,k} = v_{i,j,k} + cv_{i,j,k} \end{array}, \quad (10)$$

Each member strand has its own parameters $(\alpha_u, \alpha_v, f, \theta)$. We could use the same parameters for all member strands of a master strand. However, this would generate identical hairs which would look unrealistic. Therefore, we add random noise to given parameters for individual member strands. More noise generates more uneven hairs. Figure 7 shows some examples of curved strands that are generated using our model.

To sum up, the k-th particle of the j-th member strand of the i-th master strand has an offset position for wisp shape $(t_{i,j,k}, u_{i,j,k}, v_{i,j,k})$ and an offset position for strand shape $(cu_{i,j,k}, cv_{i,j,k})$. These are computed in advance and do not change during the simulation. Since these offset parameters are all we need in the later process, Equations (4), (6) and (9) can be replaced by more complex ones. Although Equation (10) could be computed in advance as well, we store two offsets separately for dynamic shape control.

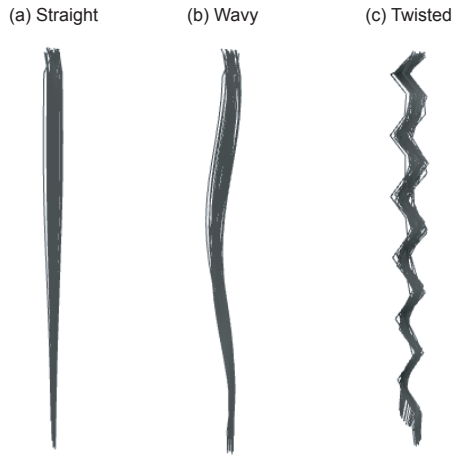

Figure 7: Example of strand shapes.

5.2 Extension to a Dynamic Model

We extend the static wisp model to a dynamic model. The offsets for wisp shapes $(u_{i,j,k}, v_{i,j,k}, t_{i,j,k})$ and for strands $(cu_{i,j,k}, cv_{i,j,k})$ are controlled based on the velocities of the corresponding particles of the master strand.

In the remainder of this section, we write $\mathbf{q}_{i,j,k} = (t_{i,j,k}, u_{i,j,k}, v_{i,j,k})$ as $\mathbf{q} = (t, u, v)$ to simplify the equations.

5.2.1 *Deformation of Wisps*

As shown in Figure 2, in general, the shape of a hair wisp becomes wider as the wisp moves faster. If we simulated all strands individually using a particle system, they would move in the same way and the size of the wisp would not change very much. However, with real human hair, the hair strands interact with each other in such a way that the movement of some strands is disturbed and they are left behind. In order to simulate this kind of wisp shape deformation, we control each particle's offset $\mathbf{q}$ in response to the velocity of the corresponding particle of the master strand (Figure 8).

However, the relationship between the velocity and the size of wisps is very complex and difficult to model, as it also depends on the material and density of the hair strands. Consequently, for the time being we simply use a linear deformation. The offset position $\mathbf{q}$ is moved based on the velocity vector $\mathbf{s}$ and a scaling parameter RS_k as shown in Figure 8.

First, we compute a unit vector $\bar{\mathbf{s}}$ which is parallel to the uv-plane by projecting the velocity vector onto the uv-plane. Next, we compute a unit vector $\bar{\mathbf{r}}$ which is perpendicular to $\bar{\mathbf{s}}$ and also parallel to the uv-plan. The horizontal offset (u, v) is then converted to (r, s) in the $\bar{\mathbf{s}}\bar{\mathbf{r}}$ coordinates,

$$\begin{pmatrix} s \\ r \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{s}} \\ \bar{\mathbf{r}} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}. \quad (11)$$

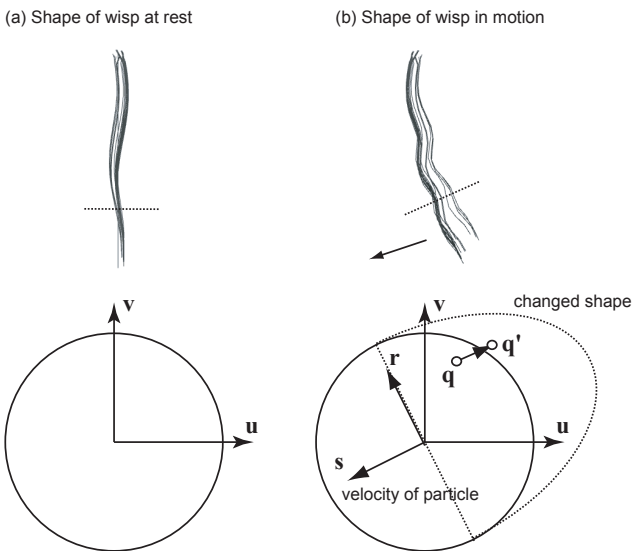

Figure 8: Control of the shape of wisp.

s is then scaled in response to the magnitude of the velocity vector $|\mathbf{s}|$ using a scaling parameter RS_k,

$$s' = \begin{array}{ll} |\mathbf{s}|\, RS_k s & if\ s < 0 \\ s & otherwise \end{array} . \tag{12}$$

In order to determine RS_k intuitively, we use the shape of the wisp $R'(t)$ at the time when the wisp moves at its fastest, $S_{\max}$, and the wisp shape is at its widest. The scaling parameter is computed from $R'(t)$ and $R(t)$,

$$RS_k = \frac{R'(k/m)}{R(k/m)\, S_{\max}}. \tag{13}$$

Since this is a linear deformation, too great a speed $|\mathbf{s}|$ may cause an unnatural shape. To avoid this, we limit $|\mathbf{s}|$ by $S_{\max}$. When $|\mathbf{s}|$ is larger than $S_{\max}$, we use $S_{\max}$ instead of $|\mathbf{s}|$ in Equation (12).

Finally, a new wisp offset of the particle $\mathbf{q} = (u', v', t)$ is computed by converting from sr-coordinates to uv-coordinates,

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{s}} & \bar{\mathbf{r}} \end{pmatrix} \begin{pmatrix} s' \\ r \end{pmatrix}. \tag{14}$$

5.2.2 *Deformation of Strands*

In addition to the shape of wisps (offsets of member strands), we control the shape of individual member strands. Based on our observation, wavy and twisted hair strands are considered to be shrunk horizontally as a result of air pressure, an example of which is shown in Figure 2.

In order to realize this kind of deformation, we introduce a linear variation model for the offsets for strand shape $(cu_{i,j,k}, cv_{i,j,k})$, which are scaled in a similar way to the offsets for wisp shape. For strand shape, we scale the offset simply in response to the magnitude of the particle velocity.

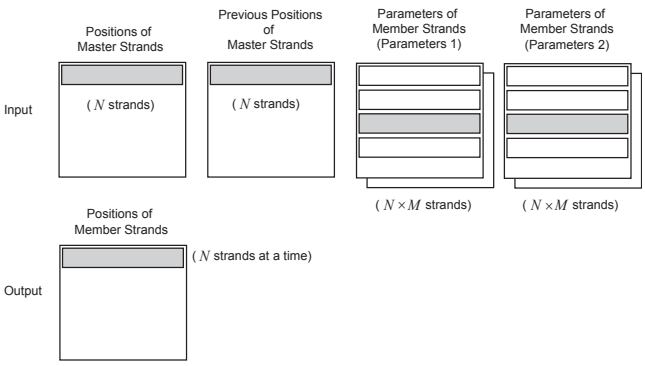

Figure 9: Textures that are used in the programmable shader for wisp generation.

Equation (10) is replaced by,

$$\begin{array}{ll} u'' = & \begin{array}{ll} u' + cu\left(1 - \frac{|\mathbf{s}|}{S_{\max}}\right) CS & if\ |\mathbf{s}| < S_{\max} \\ u' + cu \times CS & otherwise \end{array} \\ v'' = & \begin{array}{ll} v' + cv\left(1 - \frac{|\mathbf{s}|}{S_{\max}}\right) CS & if\ |\mathbf{s}| < S_{\max} \\ v' + cv \times CS & otherwise \end{array} \end{array} , \tag{15}$$

where CS is the maximum scaling factor, that plays a similar role to $R'(t)$ in Equations (12) and (13). It prevents the strand from becoming too straight.

Finally, based on $\mathbf{q}'' = (u'', v'', t)$ and Equation (8), the global position of the particle is computed. Currently our deformation changes only horizontal offsets, since vertical deformations are difficult to apply in parallel. We will discuss this issue, as well as the effectiveness of our method, in Section 6.

5.3 Implementation on GPU

Figure 9 summarizes the input and output for the shader program. A particle position of member strand $\mathbf{p}_{i,j,k}$ is computed from the position of the corresponding particle of master strand $\mathbf{p}_{i,j}$, its position in the previous frame $\mathbf{p}_{i,j}^{-1}$, offsets$(t_{i,j,k}, u_{i,j,k}, v_{i,j,k})$ $(cu_{i,j,k}, cv_{i,j,k})$, and deformation parameters $(RS_k, CS, S_{\max})$. The previous positions are used to compute the velocity $\mathbf{s}_{i,j,k}$ of each particle. We need at least two textures to store these parameters since one texture can store at most 4 (RGBA) values.

We can use either the same deformation parameters $(RS_k, CS, S_{\max})$ for all strands or different ones. In the former case, we can give the parameters $(CS, S_{\max})$ to the shader program directly instead of using a texture, although we still need two textures for five offset parameters and RS_k. The offset and deformation parameters do not change during simulation. Therefore, these textures are initialized only once at the beginning.

Note that we need these parameters for all particles of member strands, while the input positions are needed for all particles of master strands. If we have N master strands and generate M member strands from each master strand, we need a texture space of $N \times M$ strands as shown in Figure 9. In order to render $N \times M$member strands, we compute N member strands at one time and render them on the

Table 1: Computational times under various conditions.

	Conditions			Computation Times			
No	strands for simulation	strands for wisp model	particles for rendering	simulation (msec)	wisp generation (msec)	rendering (msec)	fps
1	500	10,000	100,000	4.1	9.4	2.2	47.8
2	1,000	10,000	100,000	4.5	4.1	3.1	52.2
3	1,000	20,000	200,000	4.4	6.8	7.2	38.0
4	5,000	50,000	500,000	6.4	8.2	9.9	30.3
5	20,000	20,000	200,000	18.0	1.7	6.1	28.1
6	20,000	-	200,000	18.0	-	3.9	32.0

Figure 10: Screen shots from experiments.

screen and repeat this process M times. To do this, we give the shader program the texture address for the start of the parameters of the current group of N member strands. Alternatively we could render $N \times M$ points simultaneously. In this case, we need to give the shader all textures and a large texture is required for output positions as well as an additional texture that maps texture coordinates of output particles to the texture coordinates of the input particles.

6. EXPERIMENTS AND DISCUSSION

6.1 System Implementation

We implemented the proposed method using C++ and OpenGL. We used Cg [7] for the programmable shaders.

For modelling hair styles, we implemented a simple hair style editor with which a user can specify the polygons of the head of a human figure, the length of master strands, and the parameters for dynamic wisp models using a mouse and a keyboard.

For hair rendering, we use conventional methods. When we render hairs, both the shading model of individual hair strands and the consideration of self shadows are important. The Kajiya-Kay model [10] is a popular shading model. We implemented it as a vertex shader program and it works very efficiently on a GPU. Although we have not yet implemented any self shadow models, we intend using the deep shadow method [12], which is a common way to realize self shadow and can also be implemented efficiently on a GPU [14].

6.2 Experiments

The experiments shown in this paper were tested on a standard PC (Intel Core 2 Duo E6600 2.4GHz CPU, 2 GB memory) with a video card GeForce 8600 GTS (256 MB video memory) and Windows Vista.

The computational times under various conditions are shown in Table 1. Figure 10 shows some screen shots. In all experiments, we use 10 particles for each member strand. Although the appropriate number of strands depends on a hair style and the radius of hair wisps, we need at least 10,000 to 20,000 strands to cover the skin of the figure's head. Using between 10,000 to 20,000 strands for the fine model, the animation speed was about between 40 to 50 fps (Experiments 1-3, and Figure 10 (a)-(b)), while with 50,000 strands it slowed down to about 30 fps (Experiment 4).

Computational times for the static wisp model and the dynamic wisp model are almost the same. The dynamic wisp model executes in real-time. As we expected, since the shape of wisps does not change in a static wisp model (Figure 10(c)), the dynamic wisp model (Figure 10(b)) looks more realistic.

Experiment 6 (Figure 10(d)) shows the results when only dynamic simulation is used with 200,000 particles. Comparatively it is slower than other experiments, since in dynamic simulation the positions of all particles have to be rendered on the texture several times while when generating wisp strands they are rendered only once. Moreover, when we use only dynamic simulation, no hair-hair interaction is considered, and therefore all nearby strands move uniformly. As a result, our method produces a more plausible animation.

6.3 Discussion and Future Work

As explained in Section 5, we employed a simple linear deformation model, which does not ensure physically correctness. We do not apply any constraints to generated member strands. Therefore, the length of these strands may vary during animation and they may also penetrate into the hu-

man body. However, the variations in length are usually very small and hardly noticeable. Our model takes into account only lateral velocities and ignores vertical velocities. The shape of hair wisps are considered to change horizontally and vertically in response to the vertical velocities as well. However, in order to apply vertical variations to particles, we have to propagate the variations from the base particle to the top particle along a hair strand. To do this, the process on each particle needs the result of the process on an adjacent particle. This makes the program difficult to be implemented on a GPU.

The main aim of our method is to add reality to existing real-time simulation methods with a reasonable computational cost. Physical correctness is not so important especially for online applications. The most important thing is to avoid conventional static wisps and to introduce a plausible wisp deformation. Although a more physically correct model might be possible, we think that our current model is well-suited to our purpose.

Currently, the hair style and scaling parameters have to be given manually. Future work includes developing a system for designing hair styles for our model. Developing methods for acquiring parameters from the results of dynamic simulation of complex hair styles or from real hair movements [4][16] is also an interesting extension. Moreover, the improvement of rendering (e.g. introducing self shadow as explained in Section 6.1) is also a future work.

7. CONCLUSION

In this paper, we present a real-time hair simulation technique and introduce a dynamic wisp model. The experiments show that plausible animations are achieved using a simple geometric deformation. As explained in Section 1, very few computer games or interactive applications currently employ hair simulation. No matter how human-like the characters behave, if they have solid hair, they lack reality. We expect that our approach can solve this problem and make the characters more believable.

8. REFERENCES

[1] Ken Anjyo, Yoshiaki Usami, Tsuneya Kurihara. A Simple Method for Extracting the Natural Beauty of Hair, ACM Computer Graphics (SIGGRAPH 92), Vol. 26, No. 2, pp. 111-120, 1992.

[2] Yosuke Bando, Bing-Yu Chen, Tomoyuki Nishita. Animating Hair with Loosely Connected Particles. Computer Graphics Forum (Proceedings of Eurographics 2003), Vol. 22, No. 3, pp. 411-418, 2003.

[3] F. Bertails, T.Y. Kim, M.-P. Cani, U. Neumann. Adaptive Wisp Tree - A multiresolution control structure for simulating dynamic clustering in hair motion. ACM SIGGRAPH/ Eurographics Symposium on Computer animation 2003, pp. 207-213, 2003.

[4] Kiran S. Bhat, Christopher D. Twigg, Jessica K. Hodgins, Pradeep K. Khosla, Zoran Popovi, Steven M. Seitz. Estimating Cloth Simulation Parameters from Video, ACM SIGGRAPH/ Eurographics Symposium on Computer animation 2003, pp. 37-51, 2003.

[5] Johnny T. Chang, Jingyi Jin, Yizhou Yu. A Practical Model for Hair Mutual Interactions. ACM SIGGRAPH Symposium on Computer Animation 2002, pp 73-80, July 2002.

[6] Byoungwon Choe, Hyeong-Seok Ko. A Statistical Wisp Model and Pseudo physical Approaches for Interactive Hairstyle Generation. IEEE Transactions on Visualization and Computer Graphics, Vol. 11, No.2, pp. 160-170, March 2005.

[7] Randima Fernando, Mark J. Kilgard. The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics. Addison-Wesley Professional, 2003.

[8] Rajeev Gupta, Melanie Montagnol, Pascal Volino, Nadia Magnenat-Thalmann. Optimized Framework for Real Time Hair Simulation. Computer Graphics International (CGI'06), pp. 702-710, June 2006.

[9] Sunil Hadap and Nadia Magnenat-Thalmann. Modeling Dynamic Hair as a Continuum. Computer Graphcis Forum (EUROGRAPHICS 2001), Vol. 20, No. 3, pp. 329-338, 2001.

[10] James T. Kajiya, Timothy L. Kay. Rendering Fur with Three Dimensional Textures. ACM SIGGRAPH Computer Graphics (SIGGRAPH89), Vol. 23, No. 3, pp. 271-280, 1989.

[11] Tea-Yong Kim and Ulrich Neumann, Interactive Multiresolution Hair Modeling and Editing. ACM Transactions on Graphics (Proc. of SIGGRAPH 2002), Vol 21, No. 3, pp. 620-629, 2002.

[12] Tom Lokovic, Eric Veach. Deep Shadow Maps. SIGGRAPH 2000, pp. 385-392, 2000.

[13] Mark Meyer, Gilles Debunne, Mathieu Desbrun, Alan H. Barr. Interactive Animation of Cloth-like Objects in Virtual Reality. Journal of Visualization and Computer Animation, Vol. 12, Issue 1, pp. 1-12, May 2001.

[14] Hubert Nguyen and William Donnelly. Hair Animation and Rendering in the Nalu Demo. GPU Gems 2, pp. 361-380, 2005.

[15] nVidia, GPU Cloth, http://developer. nvidia.com/, 2005.

[16] Sylvain Paris, Hector M. Briceño, Francois X. Sillion. Capture of Hair Geometry from Multiple Images. ACM Transactions on Graphics (SIGGRAPH 2004), Vol. 23, Issue 3, pp. 712-719, 2004.

[17] Pascal Volino, Nadia Magnenat-Thalmann. Real-Time Animation of Complex Hairstyles. IEEE Transactions of Visualization and Computer Graphics, Vol. 12, No. 2, pp. 131-142, March/April 2006.

[18] Rodriguez-Navarro J., M. Sainz, Susin A. Fast Cloth Simulation on Moving Humanoids. EUROGRAPHICS 2005 Short presentations, pp 85-88, 2005.

[19] Xue Dong Yang, Zhang Xu, Tao Wang, Jun Yang. The Cluster Hair Model. Graphical Models, Vol. 62, Issue 2, pp. 85-103, March 2000.

[20] Yizhou Yu. Modeling Realistic Virtual Hairstyles. Pacific Graphics 2001, pp. 295-304, 2001.

[21] Kelly Ward, Ming C. Lin, Joohi Lee, Susan Fisher, and Dean Macri. Modeling Hair Using Level-of-Detail Representations. Computer Animation and Social Agents 2003, pp. 41-47, 2003.

[22] Yasuhiko Watanabe, Yasuhito Suenaga. A Trigonal Prism-Based Method for Hair Image Generation. IEEE Computer Graphics and Applications, Vol. 12, Issue 1, pp. 47-53, January 1992.