Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2012)
J. Bender, A. Kuijper, D. W. Fellner, and É. Guérin (Editors)

Real-time Hair Simulation with Efficient Hair Style Preservation

Dongsoo Han and Takahiro Harada

Advanced Micro Devices, Inc

Abstract

Hair can be a prominent feature of characters in real-time games. In this paper, we propose hair simulation with efficient preservation of various hair styles. Bending and twisting effects are crucial to simulate curly or wavy hair. We propose local and global shape constraints and parallel methods to update local and global transforms to find goal positions. All three methods show good visual quality and take only a small fraction of rendering time. This simulation runs on the GPU and works smoothly as a part of rendering pipeline. Simulating around 20,000 strands composed of total 0.22 million vertices takes less than 1 millisecond. Simulation parameters such as stiffness or number of iterations for shape constraints can be manipulated by users interactively. It helps designers choose the right parameters for various hair styles and conditions. Also the simulation can handle various situations interactively.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Parallel

1. Introduction

Hair is an important part of a character. In recent video games, hair simulation has become crucial to achieve realistic visual rendering of a human. Unfortunately, simulating hair in a real-time environment is a non-trivial task because a human has around 100,000 hair strands and it requires heavy computation.

In this paper, we present novel methods to simulate bending and twisting effects to preserve various hair styles and support different hair conditions by using GPU. We simulate 20,000 strands consisting of 0.22 million vertices in a small fraction of rendering time.

The main contributions of this work are:

1. Parallel methods to update local and global transforms in GPU.

2. Local shape constraints to simulate bending and twisting effects in the real-time environment.

3. Global shape constraints to preserve global hair shapes and avoid getting tangled in weird shapes during fast moving gameplay.

2. Related Work

Generally, there are two approaches to represent hair: strand-based and volume-based. The strand-based method uses finite vertices and edges to represent hair. In terms of dynamics of individual hair strands, [RCT91], [SLF08] and [CCK05] use Mass-Spring systems. Stiff springs are required to simulate inextensibility of edges. Each particle has one translation and two angular rotations and hair-bending rigidity is ensured by angular springs at each joint. Due to the stiff spring, it has a numerical instability issue.

[BW98] proposed an implicit integration method that enables large time steps with stiff springs for cloth simulation. Implicit integration was later used in the case of hair simulation [WL03] [CCK05] [CCK05]. Even with stiff springs, it often requires a post-correctional process to limit the stretching of springs [IP96] [BFA02]. The implicit integration method is not suitable for real-time application due to the high computational cost of solving a linear system.

[HMT01] and [LK01] proposed methods to present hair as a rigid, multi-body, serial chain. Such techniques are well-

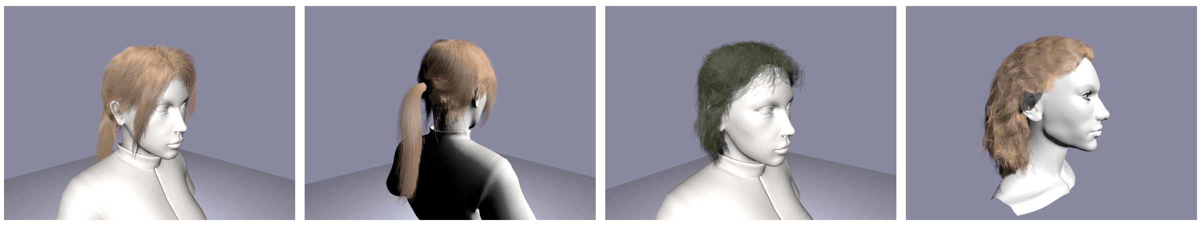


Figure 1: (a) and (b) are the same model. It has three regions(front, middle/top and ponytail) to assign different simulation parameters. (c) and (d) have wavier styles.

known in the field of robotics and efficient multi-body dynamics algorithms have been studied for a long time [Fea07].

In this paper, we use multi-body serial chain representation to maintain local coordinate systems. However we do not consider the hair as rigid links. Instead, we find the local and global goal positions for vertices in hair-strand polylines and apply shape constraints.

[RKN10] used shape matching [MHTG05] to simulate complex hair styles. [MC11] introduced oriented particles to simulate wide range of objects including ropes and could avoid possible singularity problem with polar decomposition used in shape matching. Similar to shape matching, our methods find goal positions. However, with local and global transforms and shape constraints, we avoid using polar decomposition.

In the volume-based hair representation, hair is considered as continuous medium in a global manner. [HMT01] modeled complex interactions of hair using fluid dynamics. Even with global handling of interactions of hair, individual strand dynamics are computed to capture geometry and stiffness of each hair strand. [BCN03] modeled hair using a set of SPH particles that interact in an adaptive way. [VMT04] proposed to use a global volumetric free-form deformation (FFD) scheme.

In terms of constraints, we use a position-based approach [MHHR07] for edge length and local shape constraints. Its simplicity and stability make it suitable for real-time applications with GPU implementation.

3. Simulation Overview

In this paper, we simulate all hair strands individually using a GPU. Each hair strand is represented as a polyline. As outlined in Algorithm 1, vertex and edge information of hair polylines are loaded into memory and rest-state values, such as rest lengths of edges and initial local and global transforms, are computed. When simulation begins, all data get transferred to the GPU where simulation takes place. All simulation control properties such as external forces, stiffness or head transform information are fed from CPU to GPU on a per-frame basis.

Algorithm 1: Hair simulation outline

```
1  load hair data
2  precompute rest-state values
3  **while** simulation running **do**
4      compute forces such as gravity or wind
5      integrate
6      apply global shape constraints
7      **while** iteration **do**
8          apply local shape constraints
9      apply edge length constraints
10     collision handling
```

Bending and twisting effects are simulated as global and local constraints. Both constraints find the goal positions and apply constraints to each vertex to match the goal shapes. The global constraint seeks the goal position as the initial vertex position. In the meantime, the local constraint uses a local frame to find the goal position. After applying shape constraints, we apply edge length constraints [MHHR07].

3.1. Definitions

The index of vertices starts from the root of a hair strand that is attached to a scalp. P_i is the position of vertex i in the current time step. Zeroth time step is the rest state and we use a right superscript to express it explicitly (i.e. P_i^0). In this paper, we focus only on vertices in one strand when we explain algorithms. Therefore, vertex index i is always unique.

In case we need to explicitly clarify which coordinate system we use, we specify it using left superscript (i.e., ${}^{i-1}P_i$,meaning position of vertex i in the current time step defined in the local frame *i-1*). When the position is defined in the world coordinate system, we can drop the frame index (i.e., ${}^{w}P_i = P_i$).

In terms of transforms, we define ${}^{i-1}T_i$ as a full transformation containing rotation ${}^{i-1}R_i$ and translation ${}^{i-1}L_i$. It transforms ${}^{i}P_{i+1}$ to ${}^{i-1}P_{i+1}$ such that ${}^{i-1}P_{i+1} = {}^{i-1}T_i \cdot {}^{i}P_{i+1}$. Because of careful indexing of vertices in the strand, the following equation holds:

$$^{w}T_i = {}^{w}T_0 \cdot {}^{0}T_1 \cdot {}^{1}T_2 \ldots \cdot {}^{i-2}T_{i-1} \cdot {}^{i-1}T_i \tag{1}$$

In this paper, we call $^{i-1}T_i$ local transform and $^{w}T_i$ global transform. In case of vertex *0*, local transform and global transform are the same such that $^{-1}T_0 = {}^{w}T_0$.

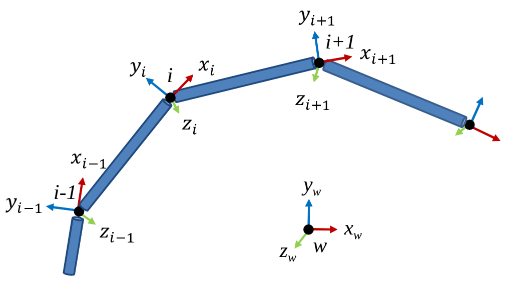


Figure 2: Local frames in a hair strand

In Figure 2, local frames are defined at each vertex. x_i, y_i and z_i are basis vectors of local frame of vertex i in the current time step. x_i is simply defined as a normalized vector of $P_i - P_{i-1}$. As an exception, $x_0 = (P_1 - P_0)\,/\,\|P_1 - P_0\|$. We will explain how to initialize and update transforms in Section 4.

To describe head transform, we use $^{w}T_H$ which transforms the head from the rest state to the current state. $^{w}T_H$ is an input from user or predefined animations.

3.2. Integration

To integrate motion of hair dynamics, we use the Verlet integration scheme because it is simple and shows good numerical stability compared to the explicit Euler method. External forces such as gravity are applied during this step. Damping coefficient is multiplied to velocity to simulate a damping effect.

We integrate only particles whose inverse mass is nonzero. If the inverse mass is zero, we consider it non-movable, update its positions following its attached objects (such as a head), and skip rest of steps for those particles. We assign zero inverse mass for vertex *0* and *1* so they can be animated following head movement.

3.3. Global Shape Constraints

Before simulation begins, we save the rest positions of vertices P_i^0. We use these rest positions as goal positions to apply global shape constraints. In Equation (2), S_G is a stiffness coefficient for the global shape constraint. It ranges between 0 and 1. If S_G is 0, there is no effect; if it is 1, the hair becomes completely rigid, frozen to the initial shape.

$$P_i \mathrel{+}= S_G({}^{w}T_H \cdot P_i^0 - P_i) \tag{2}$$

In many cases, we apply global shape constraints on a part of the hair strand such as close to root of hair. We can also gradually reduce S_G from root of hair to the end. This is because hair seems to behave more stiffly close to root; also, it maintains the hair style more efficiently without bringing unnecessary extra stiffness to overall hair simulation.

The benefit of global shape constraints is that it keeps the global shape with minimum cost. Combined with local shape constraints, it takes almost no time to settle the simulation and there is no visual perturbation when simulation starts. Designers can expect that their authored hair shape will be the initial shape. Global shape constraints also ensure that hair does not get tangled to weird shapes during fast moving gameplay.

3.4. Local Shape Constraints

Equation (3) looks similar to Equation (2) except it is written in local frame and uses local stiffness coefficient S_L. In vertex-level parallel processes, Equation (3) becomes unstable and causes excessive oscillation.

$$^{i-1}P_i \mathrel{+}= S_L({}^{i-1}P_i^0 - {}^{i-1}P_i) \tag{3}$$

Instead, we update $^{i-1}P_{i-1}$ and $^{i-1}P_i$ as in Equation (4) to achieve stable convergence.

$$\begin{aligned} {}^{i-1}d_i &= {}^{i-1}P_i^0 - {}^{i-1}P_i \\ {}^{i-1}P_{i-1} &\mathrel{-}= \frac{1}{2}S_L{}^{i-1}d_i \\ {}^{i-1}P_i &\mathrel{+}= \frac{1}{2}S_L{}^{i-1}d_i \end{aligned} \tag{4}$$

In Equation (4), we get the new positions in local frames. Eventually, we need to update their global positions using $^{w}T_i$. Based on how to update local and global transforms, we will present three methods in Section 4.

We apply local shape constraints multiple times at each frame. User's input can control the number of iterations. If the hair style is very curly, then more iteration is usually needed. In our method, we do not distinguish bending and twisting effects but it would be possible. We simply combine them for the sake of performance and simplicity of real-time application.

If we apply the local shape constraints with very large number of iterations, the resulting effect would be the same as the global shape constraints because both shape constraints will converge to the rest shape when S_G and S_L are 1. However, because we usually use the small number of iterations and apply the global shape constraints with variable S_G, both show different effects. With global shape constraints,

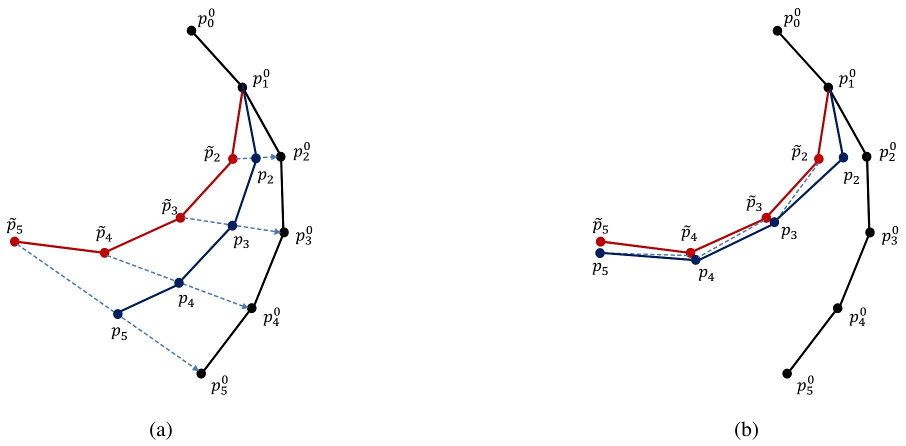


Figure 3: (a) illustrates global shape constraints and (b) illustrates local shape constraints after one iteration. To be simple, head transform is not considered and (b) is based on Equation (3). P_i^0 is rest position and $\tilde{P}_i$ is intermediate position. $S_G = S_L = 0.5$. Note that P_0 and P_1 are not simulated.

we need less iterations for local shape constraints and users have more controls over simulation. Figure 3 shows rest positions and updated positions after applying global and local shape constraints with one iteration.

Compared to shape matching [RKN10] [MHTG05], our local shape constraint method does not require polar decomposition and overlapping regions. Therefore, it can exploit GPU parallelism more effectively.

3.5. Collision Handling

In Algorithm 1, collision handling is the last step. We use simple collision objects such as a sphere to represent character head and shoulders. We do not handle any inter-collisions between hair and hair.

If a hair vertex is detected to have penetrated collision objects, we simply move it to the closest surface point and update its position.

4. Local and Global Transforms

Initializing and updating local and global transforms are the most crucial part of this paper.

4.1. Initialize Local and Global Transforms

Initializing local and global transforms is performed when rest-state values are computed. In Algorithm 2, we start with vertex *0* and subsequently compute the next vertex in order. For vertex *0*, we explicitly compute basis vectors x_0^0, y_0^0, and z_0^0. As explained in Section 3.1, x_0^0 is a normalized vector of $P_1^0 - P_0^0$. We choose y_0^0 by cross-product x_0^0 with an arbitrary vector. If the arbitrary vector is coincident with x_0^0, we simply choose another arbitrary vector that is orthogonal to the previous choice.

Algorithm 2: Initialize local and global transforms

```
1  for each strand in total hair
2      for i ← 0 to number of vertices in strand
3          if i is 0 then
4              compute basis vectors ($x_0^0$, $y_0^0$, $z_0^0$)
5              initialize ${}^wT_0^0$ from basis vectors and $P_0^0$
6          else
7              compute ${}^{i-1}x_i^0$
8              initialize local transform ${}^{i-1}T_i^0$ by rotation from
               vector (1, 0, 0) to ${}^{i-1}x_i^0$
9              initialize global transform ${}^wT_i^0$
```

For vertices whose indices are greater than zero, we compute ${}^{i-1}x_i^0$ first. Since it needs to use ${}^wT_{i-1}^0$, the initialization should be done in order from vertex *0*. We initialize the local transform ${}^{i-1}T_i^0$ with rotation from vector (1, 0, 0) to ${}^{i-1}x_i^0$ and translation p_i^0. Finally we initialize ${}^wT_i^0$ using Equation 1.

4.2. Update Local and Global Transforms

Updating local and global transforms is necessary to apply local shape constraints. This process is similar to the initialization and each computation requires to use the result from the previous one, which makes it serial process. In this section, we present three methods. Method 1 is a serial process within a strand. Method 2 is fully parallel in the vertex level. Method 3 skips several key steps in the method 2 but produces good visual quality.

In method 1 presented in Algorithm 3, updating local and

Algorithm 3: Update local and global transforms (Method 1)

```
1   set number of iterations
2   for iteration ← 0 to number of iterations
3       for each strand in total hair
4           for i ← 0 to number of vertices in strand
5               if i is 0 or 1 then
6                   update $^{w}T_0$ and $^{w}T_1$ and continue
7               else
8                   compute $^{i-1}d_i$
9                   apply a constraint to $p_{i-1}$ and $p_i$ using $^{i-1}d_i$, $^{w}T_{i-1}$ and $^{w}T_H$
10                  compute $^{i-1}x_i$
11                  update $^{i-1}T_i$ by rotation from vector (1, 0, 0) to $^{i-1}x_i$
12                  update $^{w}T_i$ using $^{w}T_{i-1}$ and $^{i-1}T_i$
```

global transforms is performed in the order of vertex index basically as in the initialization method in Algorithm 2. When a local constraint is applied to vertex i, we eventually need to use $^{w}T_i$. Because of the serial dependency as in Equation (1), the process should be serial and it prohibits from exploiting massive parallel threads in GPU architecture.

However, since all local and global transforms in method 1 are always accurate, the simulation quality is very nice and we use the method 1 as a standard to check the performance and quality of other methods.

In method 2 presented in Algorithm 4, we use $^{w}T_{temp_i}$ to decouple the dependency. It is basically Jacobi-style iteration. This method is fully parallel in vertex-level. In line 6 of Algorithm 4, threads synchronization is placed so all threads can use the latest $^{w}T_{temp_i}$.

Algorithm 4: Update local and global transforms (Method 2)

```
1   set number of iterations
2   for iteration ← 0 to number of iterations
3       for each strand in total hair
4           for i ← 0 to number of vertices in strand
5               $^{w}T_{temp_i} \leftarrow {}^{w}T_i$
6       sync threads
7       for each strand in total hair
8           for i ← 0 to number of vertices in strand
9               if i is 0 or 1 then
10                  update $^{w}T_0$ and $^{w}T_1$ and continue
11              else
12                  compute $^{i-1}d_i$
13                  apply a constraint to $p_{i-1}$ and $p_i$ using $^{i-1}d_i$, $^{w}T_{temp_{i-1}}$ and $^{w}T_H$
14                  compute$^{i-1}x_i$
15                  update $^{i-1}T_i$ by rotation from vector (1, 0, 0) to $^{i-1}x_i$
16                  update $^{w}T_i$ using $^{w}T_{temp_{i-1}}$ and $^{i-1}T_i$
```

In method 3 presented in Algorithm 5, we skip updating local and global transforms and reuse the rest-state transforms. Compared to method 1 and 2, the updated position can be wrong in the first iteration. However it is correct for vertex *2* which is the first vertex simulated in strand. In a

Algorithm 5: Update local and global transforms (Method 3)

```
1   set number of iterations
2   for iteration ← 0 to number of iterations
3       for each strand in total hair
4           for i ← 0 to number of vertices in strand
5               compute $^{i-1}d_i$
6               apply a constraint to $p_{i-1}$ and $p_i$ using $^{i-1}d_i$, $^{w}T^{0}_{i-1}$ and $^{w}T_H$
```

few iterations, the updated position becomes correct gradually because the local and global transforms get corrected by propagation from vertex *0*. Thanks to the position-based dynamics, we do not accumulate any forces. We only update the positions and intermediate positions would not matter as long as the final position update is correct.

Because we set the number of iterations as a small number such as 3 or 4, method 3 may not converge to method 1 and local and global transforms may be incorrect. However, the visual artifact is not much noticeable especially in the fast moving animation.

5. GPU Implementation

We use compute shader in DirectX 11 to run hair simulation in the GPU. Each thread group takes care of one hair strand and each thread in the thread group handles each vertex or a group of vertices in the same batch.

For local shape and edge length constraints, batching is necessary because we need to avoid updating the same positions from the different threads. For a complex mesh, batching becomes complicated; it is closely related to the coloring problem. Thanks to the simple structure of hair polyline, we can simply create batches by grouping vertices following the order of vertex indices.

Initially we pass information such as vertex position, local and global transforms and edge rest lengths to global memory in the GPU. When thread kernel gets called, it loads that information into shared memory to take advantage of fast access to shared memory.

In Figures 1a and 1b, hair strands are grouped into three regions (front, middle/top, and ponytail). Each region is assigned different simulation parameters such as stiffness or number of iterations for local shape constraints. Having regions helps designers have more subtle controls. It is possible to have more regions without any impact on GPU performance.

6. Results and Conclusion

We have presented three methods to update local and global transforms. As we can see in Figure 5, those methods produce visually satisfying simulation results. With local and global transforms, we apply local and global shape constraints and simulate various hair styles in massive parallel environment.

Table 2 shows simulation results tested on an AMD Phenom^{TM}II X4 955 processor running at 3.21 GHz and AMD Radeon^{TM}HD 7970 and AMD FirePro^{TM}W8000 GPU cards. With the same simulation parameters, method 3 is 5 to 29 times faster than method 1. In terms of visual quality, method 1 shows the smoothest simulation result. Method 2 can be as smooth as method 1 if a high number of iterations is used. In our experiments, we used small iterations. As a result, the local and global transforms are not as accurate as in method 1 and the visual quality is lower than that. Method 3 shows some jagged strands but is acceptable for real-time games.

With simulation time results and visual qualities, it is possible to support LOD-style simulation. We can choose method 1 for close-up scenes and method 2 or 3 for distant scenes.

In Table 2, model 3 needs more iterations for edge length and local shape constraints because its hair style is wavier than other models. Also it has many more vertices. Those are the reasons the simulation takes more time than other models.

Figure 4 shows the user interface that allows the user to change simulation and rendering parameters interactively. By controlling stiffness, global constraint range, number of iterations and damping, various hair conditions such as heavy or wet can be simulated as in Figure 6.

For future work, we would like to add inter-collision handling to achieve more realistic simulation. As an extension, fur or vegetation will be interesting research topics.

7. Acknowledgments

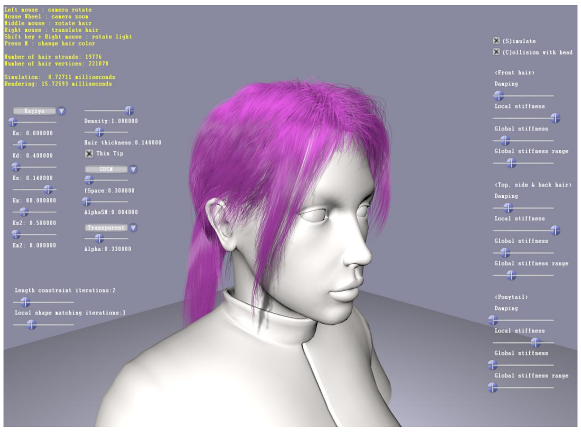
We thank Timothy Heath for hair modeling and Karl Hillesland and Sujeong Kim for helpful discussion and comments. Also we appreciate Joe Alter for allowing us to use Shave and Haircut software for content creation.

	Strands	Total vertices	Collision objects
Model 1	19,776	221,078	3 spheres
Model 2	20,000	259,993	3 spheres
Model 3	20,000	1,373,047	3 spheres

Table 1: Model 1 is Figures 1a and 1b. Model 2 is Figure 1c. Model 3 is Figure 1d.

		Edge length constraint iterations	Local shape constraint iterations	Ave. simulation time per frame (milliseconds)	
				AMD Radeon HD 7970	AMD FirePro W8000
Model 1	Method1	2	3	4.59	5.32
	Method2	2	3	1.22	1.44
	Method3	2	3	0.78	0.94
Model 2	Method1	3	3	5.32	6.28
	Method2	2	3	1.24	1.46
	Method3	2	3	0.80	0.95
Model 3	Method1	3	4	62.38	73.84
	Method2	3	4	3.21	3.81
	Method3	3	4	2.11	2.51

Table 2: Simulation results


```
Left mouse : camera rotate
Mouse Wheel : camera zoom
Middle mouse : rotate hair
Right mouse : translate hair
Shift key + Right mouse : rotate light
Press M : change hair color

Number of hair strands: 19776
Number of hair vertices: 221078

Simulation:  0.72711 milliseconds
Rendering: 15.72593 milliseconds

Kajiya
Ka: 0.000000
Kd: 0.400000
Ks: 0.140000
Ex: 80.000000
Ks2: 0.500000
Ex2: 8.000000

Density:1.000000
Hair thickness:0.140000
[✓ Thin Tip]
SDSM
fSpace:0.300000
AlphaSM:0.004000
Transparent
Alpha:0.330000

Length constraint iterations:2
Local shape matching iterations:3

[✓ (S)imulate]
[✓ (C)ollision with head]
<Front hair>
Damping
Local stiffness
Global stiffness
Global stiffness range
<Top, side & back hair>
Damping
Local stiffness
Global stiffness
Global stiffness range
<Ponytail>
Damping
Local stiffness
Global stiffness
Global stiffness range
```

Figure 4: User interface

References

[BCN03] Bando Y., Chen B.-Y., Nishita T.: Animating hair with loosely connected particles. *Computer Graphics Forum 22*, 3 (2003), 411–418. 2

[BFA02] Bridson R., Fedkiw R., Anderson J.: Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 594–603. 1

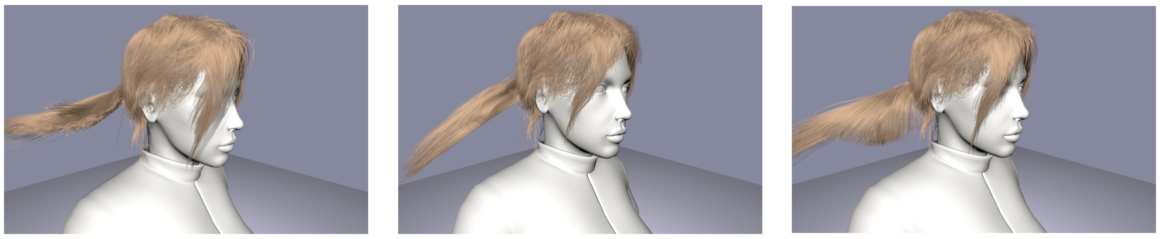
[BW98] Baraff D., Witkin A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. 1

[CCK05] Choe B., Choi M. G., Ko H.-S.: Simulating complex hair with robust collision handling. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 153–160. 1

[Fea07] Featherstone R.: *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. 2

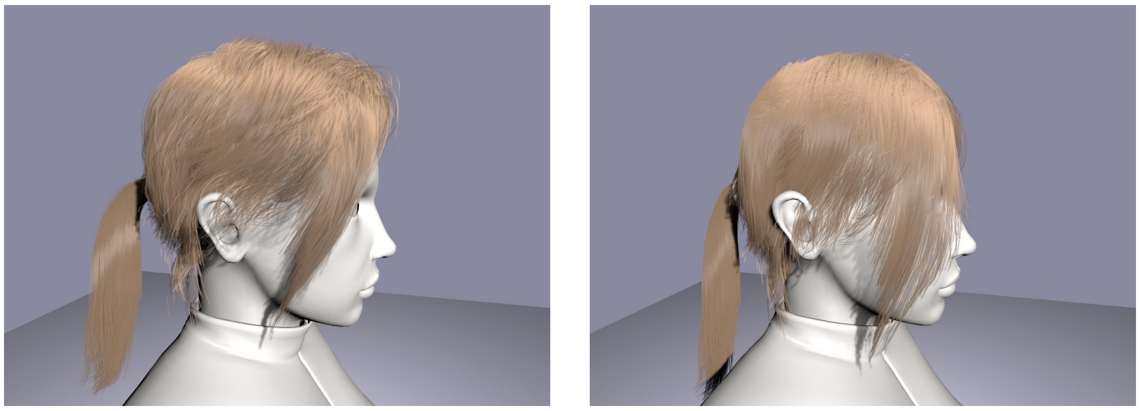
[HMT01] Hadap S., Magnenat-Thalmann N.: Modeling dynamic hair as a continuum. *Computer Graphics Forum 20*, 3 (2001), 329–338. 1, 2

[IP96] Institut X. P., Provot X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *In Graphics Interface* (1996), pp. 147–154. 1



(a) Method 1 (b) Method 2 (c) Method 3

Figure 5: Comparing methods 1, 2, and 3



(a) (b)

Figure 6: Interactive simulation of different hair conditions

[LK01] Lee D.-W., Ko H.-S.: Natural hairstyle modeling and animation. *Graphical Models 63*, 2 (2001), 67 – 85. 1

[MC11] Müller M., Chentanez N.: Solid simulation with oriented particles. In *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 92:1–92:10. 2

[MHHR07] Müller M., Heidelberger B., Hennix M., Ratcliff J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. 2

[MHTG05] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 471–478. 2, 4

[RCT91] Rosenblum R. E., Carlson W. E., Tripp E.: Simulating the structure and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation 2*, 4 (1991), 141–148. 1

[RKN10] Rungjiratananon W., Kanamori Y., Nishita T.: Chain shape matching for simulating complex hairstyles. *Comput. Graph. Forum* (2010), 2438–2446. 2, 4

[SLF08] Selle A., Lentine M., Fedkiw R.: A mass spring model for hair simulation. In *ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 64:1–64:11. 1

[VMT04] Volino P., Magnenat-Thalmann N.: Animating complex hairstyles in real-time. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2004), VRST '04, ACM, pp. 41–48. 2

[WL03] Ward K., Lin M.: Adaptive grouping and subdivision for simulating hair dynamics. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on* (oct. 2003), pp. 234 – 243. 1