REAL-TIME HAIR SIMULATION AND RENDERING

By

YANG CHEN

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2014

To my family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Jörg Peters for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. I would also like to thank my supervisory committee members, Dr. Benjamin Lok and Dr. Alper Üngör for their assistance and guidance. Special thanks to my family and friends for their lifetime support, encouragement, and love.

TABLE OF CONTENTS

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

REAL-TIME HAIR SIMULATION AND RENDERING

By

Yang Chen

May 2014

Chair: Jörg Peters
Major: Digital Arts and Sciences

Hair modeling is a fundamental part of creating virtual humans in computer graphics. Due to the complexity of human hair, realistically representing hair in structure, motion and visual appearance offers one of the most challenging physics and rendering problems, especially in real-time.

With recent advancements in both graphics hardware and software methods, real-time hair rendering and simulation are now possible with reasonable performance and quality. In this thesis, we presented a complete framework of real-time hair modeling. The framework encompasses hairstyling, hair simulation, and hair rendering.

CHAPTER 1
INTRODUCTION

Hair modeling is an extremely important component for visual characters. However, in real-time applications, it is usually avoided or extremely simplified. A person can have over hundred thousand hair strands and each individual hair strand is a thin fiber and can form complicated shapes. Because of the complexity of hair and lack of comprehensive physic model, hair modeling offers one of the most challenging simulation and rendering problems for professional computer graphics.

According to Magnenat-Thalmann et al. [32], hair modeling can be divided into three general categories: hairstyling, hair simulation, and hair rendering. Hairstyling, viewed as modeling the shape of the hair, incorporates the geometry of the hair and specifies the density, distribution, and orientation of hair strands. Hair simulation involves the dynamic motion of hair, including collision detection between the hair and objects, such as the head or body, as well as hair mutual interactions. Finally, hair rendering entails color, shadows, light scattering effects, transparency, and anti-aliasing issues related to the visual depiction of hair on the screen.

Utilizing the massive parallel computing power of the computer graphics hardware, we demonstrate the possibility of real-time hair modeling by contributing a complete framework integrating hairstyling, hair simulation, and hair rendering.

The goal of this thesis work is to add hair modeling to the existing iPASS (interactive pixel-accurate shading of surfaces [54]) project. This project aims to render the open source movie “Elephant's Dream” in real-time.

CHAPTER 2
BACKGROUND

2.1 Hairstyling

The high geometric complexity of hair and the wide variety of real world hair styles make hair modeling a challenging task. Therefore, most hair modeling techniques are based on controlling collections of hair strands at once.

A common approach to represent groups of strands is using two-dimensional parametric surfaces (e.g. NURBS) called strips [25] [28] [33]. These surfaces look like a flat group of hair strands by using texture mapping with alpha channels.

Different physically-based techniques also have been used to shape hair strands. Anjyo et al. [24] simulated the effect of gravity to find the rest poses of hair strands. Hadap and Magnenat-Thalmann [4] modeled hairs as streamlines from a fluid dynamics simulation around the head. While various hair types can be modeled with these approaches, just like other simulation methods, they can be difficult to control in a precise manner.

Capturing a hair model from images [6] [26] [15] [36] is another alternative used to automate the virtual hair modeling process. Shortage of these approaches is the lack of any artistic control.

Sketch based interfaces are also used for modeling hair by Malik [37], both for cartoon hairstyles by Mao et al. [29] and more realistic models by Wither et al. [30]. Recently, Fu et al. [49] proposed a sketch based interface to build a vector field, which is then used to generate individual hair strands. While these techniques are practical for quickly generating a hairstyle, they are very difficult to control for achieving a desired outcome precisely.

Yuksel et al. [23] presents hair meshes that aim to bring hair modeling as close as possible to modeling polygonal surfaces. This new approach provides artists with direct control of the overall shape of the hair, giving them the ability to model the exact hair shape they desire. They use the hair mesh structure for modeling the hair volume with topological constraints that allow automatically and uniquely trace the path of individual hair strands.

2.2 Hair Simulation

Unlike solids or fluids, hair modeling lacks a comprehensive physical model for accurate simulation of hair motion. Animation of a full head of hair raises obvious problems in terms of computational costs due to the complexity of hair. As a consequence, existing hair animation methods propose a tradeoff between realism and efficiency, depending on the intended application.

2.2.1 Dynamics of Individual Hair Strands

Within the last 20 years, three families of computational models have been proposed and used for simulating the dynamics of one individual hair strand: mass-spring systems, projective dynamics, and rigid multi-body serial chains. Very recently, some existing work on static Kirchhoff rods [10] [35] has been extended to hair dynamics, leading to a new model called dynamic Super-Helices.

First attempt to animate individual hair strands is the Mass-spring systems presented by Rosenblum et al. [39] in 1991. A single hair strand is treated as a set of particles connected with stiff springs and hinges. Other approaches [8], [12] used a constrained mass-spring model, well-suited for animating extensible wisps such as wavy or curly wisps.

In 1992, Anjyo et al. [23] proposed a simple method based on one-dimensional projective differential equations for simulating the dynamics of individual hair strands. It easily simulates tens of thousands of hair strands efficiently. However, this method cannot properly handle fully 3D hair motions because of the lack of torsional hair stiffness. Furthermore, as motion is processed from top to bottom, it is difficult to handle external punctual forces properly.

In order to compute the motion of individual hair strands, forward kinematics have been used [40], [22]. Such techniques are well known in the field of robotics, and efficient multi-body dynamics algorithms have been proposed for decades [38]. Each hair strand can be represented as a serial, rigid, multi-body open chain using the reduced or spatial coordinates formulation [38]. Results for these methods have typically been limited to straight hair as possible issues related to curly hair simulation are not explained.

Bertails et al. [11] exploited the Kirchhoff's theory for elastic rods to predict the motion of individual hair strand. The resulting mechanical model for one individual hair strand, called a Super-Helix, corresponds to a spatial discretization of the original continuous Kirchhoff model, where the curvatures and the twist of the rod are assumed to remain constant over each predefined piece of the rod. As a result, the hair strand is constructed as a piecewise helix, with a finite number of degrees of freedom. This model is then animated using the principles of Lagrangian mechanics. The super-Helix model naturally accounts for the typical nonlinear behavior of hair, as well as for its bending and twisting deformation modes. Finally, unlike all previous models, hair natural curliness is properly handled through this model, making it possible to accurately simulate the dynamics of curls.

2.2.2 Simulating the Dynamics of a Full Hairstyle

The realism of the collective dynamic behavior and the efficiency of the simulation are two major challenges in the simulation of a full hairstyle. Hair is essentially either globally considered as a continuous medium, or as a set of disjoint groups of hair strands.

Hadap and Magnenat-Thalmann [40] simulated the complex interactions of hair using fluid dynamics by considering hair as a continuum. Bando et al. [51] have modeled hair using a set of SPH particles that interact in an adaptive way. Chang et al. [22] created a system to capture the complex interactions that occur among hair strands. In this work, a sparse hair model of guide strands, which were first introduced in [1], [43], is simulated. A dense hair model is created by interpolating the position of the remaining strands from the sparse set of guide strands.

In order to reduce the complexity of hair, an alternative approach consists of grouping nearby hair strands and simulating these disjoint groups as independent, interacting entities. The complexity of hair simulation has been simplified by modeling groups of strands using a thin flat patch, referred to as a strip [34][44][5][52][45][9][18]. A simple dynamics model for simulating strips presented in [5] that is adapted from the projective angular dynamics method introduced by Anjyo et al. [23]. Dynamics is applied to the control point mesh of the NURBS surface. One of the first methods to take advantage of grouping hair was presented by Watanabe and Suenaga in [53]. They animate a set of trigonal prism-based wisps.

2.3 Hair Rendering

Realistic rendering of human hair requires the handling of both local and global hair properties. To render a full hairstyle, it is necessary to choose an appropriate global representation for hair. Global hair properties also include the way in how hair fibers cast shadows on each other.

2.3.1 Hair Representation

Hair can be represented either explicitly or implicitly. The possible choices for a hair rendering algorithm depends largely on the underlying representation used for modelling the geometry of the hair. Explicit representation for the hair, for example, represents an individual hair by a curved cylinder. The early work by Watanabe and Suenaga [53] adopted a trigonal prism representation. Other approaches to represent the hair use a continuous strip of triangles, where each triangle is always oriented to face the camera. When rendering an explicit representation, aliasing problems may occur because of the thinness of hair strands. Implicitly representation, for example, Kajiya, et al. [21] used volumetric textures (or texels) to avoid the problem of aliasing by using pre-filtered shading features. The cost of ray traversal is relatively low for short hairs, but can be high for long hairs. Such volumes should be updated for every frame when hair animates, making pre-filtering inefficient.

2.3.2 Hair Illumination

To make a human hair look realistic, it is necessary to take into account both the optical properties of each hair fiber, which belong to the local properties of illumination, and the light interactions occurring between the hairs, which belong to the global properties. Local hair properties define the way in which individual hair fibers are illuminated, while global hair properties also include the way in how the fibers cast shadows on each other. Self-shadowing is particularly important for creating a volumetric appearance of hair.

The two most popular local illumination models used in hair rendering are Kajiya & Kay [21], and Marschner [41]. Kajiya and Kay's model includes a diffuse component and a specular component. Kajiya and Kay derived the diffuse component by integrating reflected radiance across the width of an opaque, diffuse cylinder. Their specular component is simply motivated by the argument from the preceding section that the ideal specular reflection from the surface will be confined to a cone and therefore the reflection from a non-ideal fiber should be a lobe concentrated near that cone. Marschner proposed the most complete physically-based hair scattering model to date. Their model makes two improvements to Kajiya and Kay's model: it predicts the azimuthal variation in scattered light based on the ray optics of a cylinder, and it accounts for the longitudinal separation of the highlight into surface-reflection, transmission, and internal-reflection components that emerge at different angles.

Hair fibers cast shadows onto each other, as well as receiving and casting shadows from and to other objects in the scene. Self-shadowing creates crucial visual patterns that distinguish one hairstyle from another. Two main techniques are generally used to cast self-shadows into volumetric objects: ray casting through volumetric densities and shadow maps.

Ray casting: with implicit hair representations, one can directly ray trace volume density [50], or use two-pass shadowing schemes for volume density [21]; the first pass fills volume density with shadow.

Shadow Maps: LeBlanc [2] introduced the use of the shadow map, a depth image of hair rendered from the light's point of view. In this technique, hair and other objects are rendered from the light's point of view and the depth values are stored. Each point to be shadowed is projected onto the light's camera and the point's depth is checked against the depth in the shadow map.

Global Illumination Models can roughly be divided into two broad categories. These are Path Tracing and Photon Mapping. The most accurate global illumination model for hair would be to use brute-force Monte Carlo Path Tracing. However, the rate of convergence of this algorithm makes it a prohibited slow method. Kajiya, et al. [21], Gupta, et al. [50], and Yuksel, et al. [57] are three examples of papers that use Path Tracing for illuminating the hair. Photon mapping methods have proven successful for hair rendering. Photon mapping can smoothly approximate multiple scattering in volumes by estimating densities of traced particles. However, those approaches are memory intensive and still require several hours of computation to generate high quality still images. Moon, et al. [31] and Zinke, et al [59] are two examples of papers that use Photon Mapping for illuminating the hair.

2.4 DirectX 11

DirectX 11, which is the latest Microsoft's graphics API, introduces the Tessellation stage: the hull shader, the tessellation engine, and the domain shader.

These new tessellation stages enable programmable hardware tessellation on the modern graphics hardware. Tessellation uses the GPU to calculate a more detailed surface from a low-detail surface constructed from quad patches, triangle patches or isolines. To approximate the higher-order surfaces, each patch is subdivided into triangles, points, or lines based on tessellation factors.

The hull shader is a programmable shader that produces output control points directly to the domain shader from input control points passed by the vertex shader. In this stage we can compute any per patch attributes, transform the input control points to a different basis, and compute tessellation factors. Tessellation factors are floating point values which tell the hardware how many new vertices you would like to create for each patch, and are necessary outputs of the Hull Shader.

The next stage is the Tessellator, which is fixed function. It tessellates a domain (quadrilateral, triangle, or line) into many smaller objects (triangles, points or lines). Note that the Tessellator does not actually create any vertex data – this data has to be calculated by the programmer in the Domain Shader.

The domain shader is a programmable shader that evaluates the vertex position in the output patch. The Domain Shader is run once for each final vertex. In this stage we get as input the parametric uvw coordinates of the surface, along with any control point data passed from the Hull shader. Using these inputs we can calculate the attributes of the final vertex. After the domain shader completes, tessellation is finished and pipeline data continues to the next pipeline stage.

In addition to these new tessellation stages, DirectX 11 introduces the compute Shader as a way to utilize the computational resources on GPU without so many constraints. As a programmable stage, it enables more general algorithms far beyond shading by providing high-speed general purpose computing and taking advantage of the large number of parallel processors on the GPU. The compute shader supports more effective parallel programming methods with memory sharing and thread synchronization features similar with other general purpose GPU techniques.

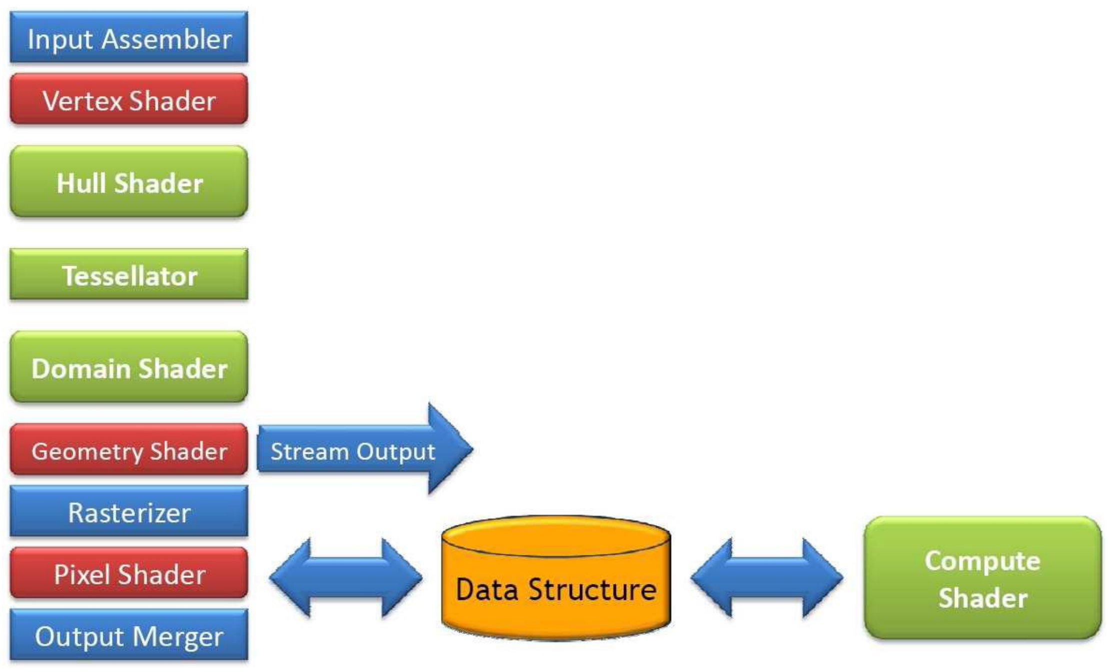

Figure 2-1. The DirectX 11 pipeline [14]

CHAPTER 3
IMPLEMENTATION DETAILS

We use the approach of [58] to create hair strands from the polygon mesh created using modeling software such as 3ds Max or Maya. Then we simulate a certain number of guide strands to guide motion of the full hair. A dense hair model is created by interpolating the position of the remaining strands from the sparse set of guide strands [22]. The final result is rendered by using the model of Kajiya & Kay [21].

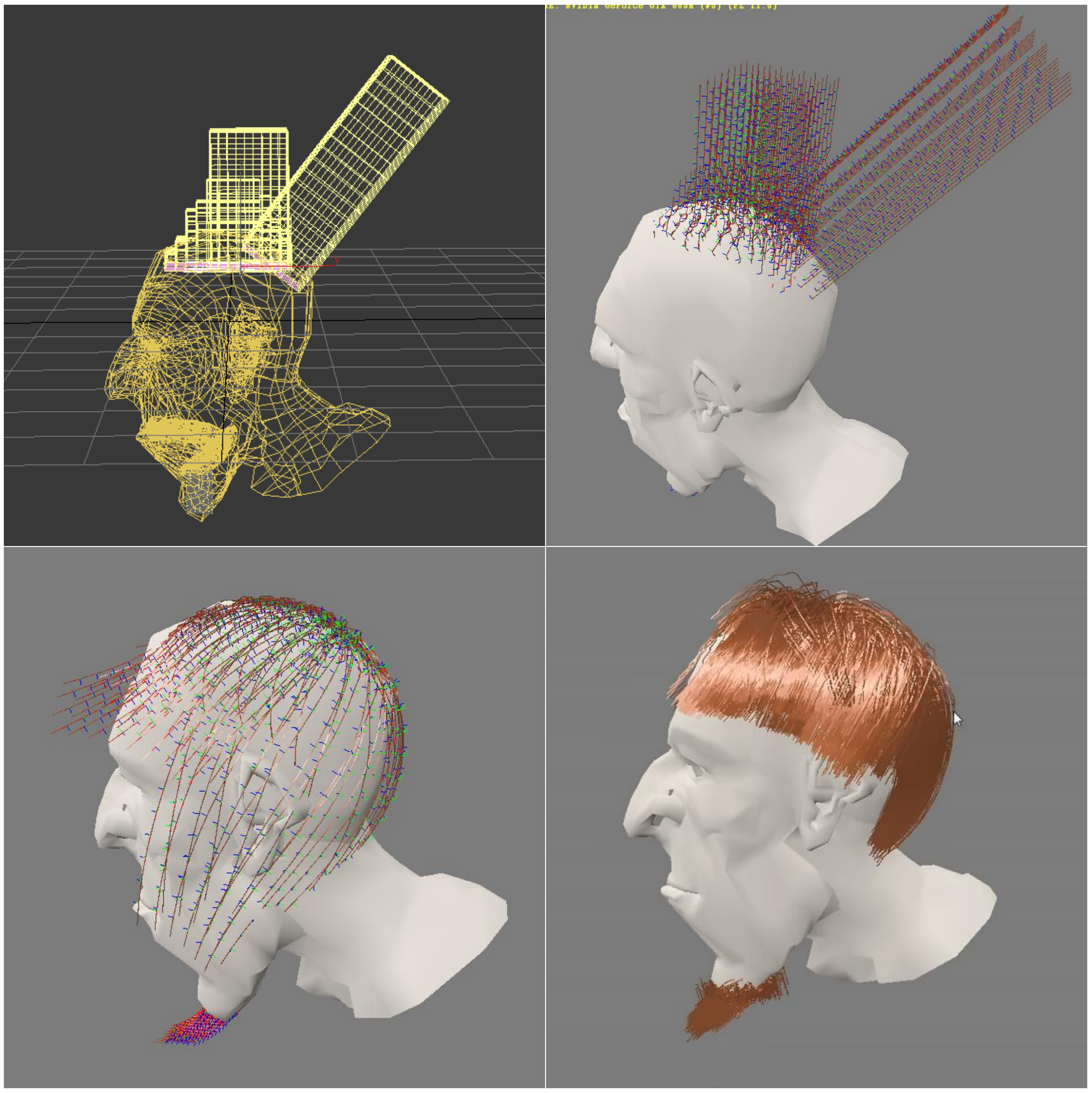

Figure 3-1. Overview of our framework implementation

3.1 Hairstyling

The hair meshes [58] allow the users or artists to easily build the shape of hair by creating polygon models. First we build the hair model using modeling software like 3dmax or Maya. Then the hair mesh is exported to our program to generate guide strands for later simulation stage.

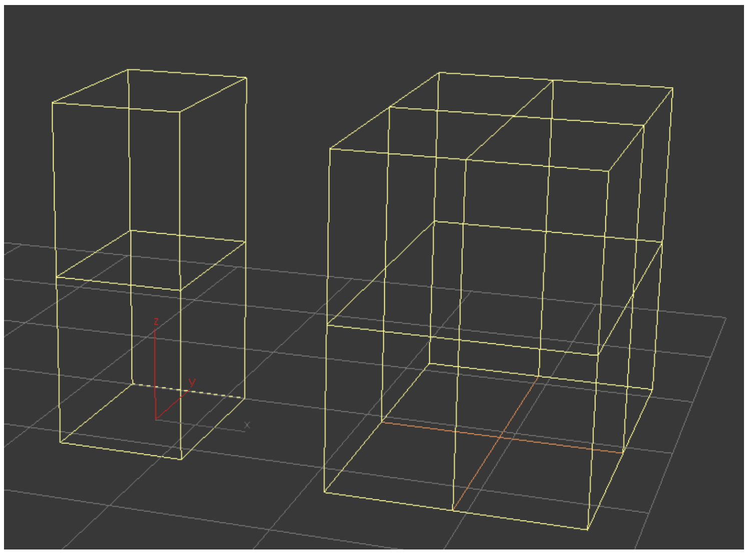
Figure 3-2. Left is a single volume of hair mesh; right has four volume hair meshes

A hair mesh model is a 3D mesh describing the volume of space enclosing the hair. It consists of a certain number of single volume hair meshes. Each volume of hair meshes is consisted of a number of layers. Each layer is represented by one polygonal face. Let F_K be the face for layer K. Then F_0 will be the root layer of the hair mesh. The root layer could simply be the polygonal face of the scalp model. We then create a newer layer F_1 by extruding the face from the root layer. By extruding the layers one by one, we create a single strand of hair mesh. The last layer is called the tip layer. Each face $F_{K,j}$ at layer K has a one-to-one correspondence to a face $F_{K+1,j}$ at the next layer. Connecting the two corresponding faces $F_{K,j}$ and $F_{K+1,j}$, we form a prism such that these faces are the two base faces of the prism. The single hair mesh is a collection of such prisms starting at root face and connecting all the faces until tip face. This collection of such prisms is called a volume of hair meshes. The users or artists could create any shape of hairstyle by creating different hair meshes and manipulating the position of vertices.

The hair mesh model is then exported to our program to generate the guide hair strands for our simulation stage. Given this correspondence between layers, we can create a path for each hair from the root layer. As a reminder, our single guide strand for simulation model is consisted of a list of vertices. For each polygonal face on the root layer, we compute the barycentric coordinates of the four face vertices to get the position of our hair strand vertex. After we trace the path of hair mesh from the root layer to the tip layer, we calculate all the positions of our guide strand vertices. By tracing all the hair meshes, we have all the guide strands for later simulation usage. We can always subdivide the layers by creating more hair strand vertices using tools inside modeling software before mesh is exported. We can also subdivide the polygonal faces to generate more hair meshes resulting in more guide strands for our hair model.

3.2 Hair Simulation

Because of the huge number of human hair, it’s impossible to simulate all individual hair in real-time. An appropriate approach is to simulate a certain number of guide strands. These guide strands define the motion of full head of hair. A dense hair model is created by interpolating the position of the remaining strands from the sparse set of guide strands [22].

3.2.1 Guide Strands Simulation

We simulate a number of guide strands created in last hairstyling stages to guide the motion of full hair using compute shader. Then we take advantage of tessellation stage to interpolate sparse set of guide strands to create a dense model of hair.

We simulate the guide strands using the Mass-Spring Systems model. Each single guide strand is represented as a chain of particles connected by stiff longitudinal springs and angular springs called hinges. And every particle has a mass and three degrees of freedom, one in translation and two in rotation. Angular springs model the stiffness of the hair bending.

We explore the multi-tasking power of the GPU by assigning each guide strand to a group of threads. Then we can have every vertex on guide strand simulated on one single thread. Hence we could simulate all the hair vertices in parallel. All the hair vertices are simulated as particles. Links between hair vertices are treated as distance constraints which maintain the length of the hair, preventing it from stretching or compressing. A distance constraint between two particles is enforced by moving them away or towards each other so that they are at exactly a pre-specified distances apart (Figure 3-3). Angular constraints and stiffness of hair strand help maintain the shape of the hair by restricting the angle between two bending segments within a certain value (Figure 3-4). Collision constraints keep the hair outside of collision obstacles. A simple sphere collision is implemented to prevent hair strands penetrate through the head model (Figure 3-5). The penalty collision energy can be calculated with the following equation:

$$F_{penalty} = \frac{1}{2}k(\max(dent, 0))^2$$

where k is the stiffness parameter of the collision geometry.

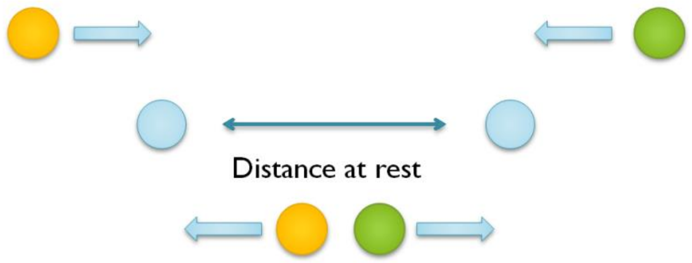

Figure 3-3. Distance constraints

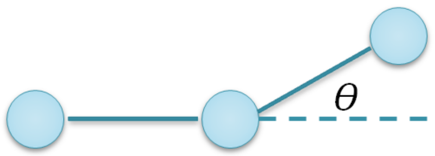

Figure 3-4. Angular constraints and stiffness

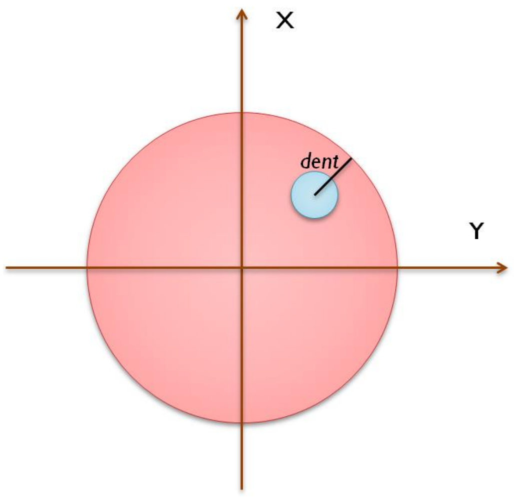

Figure 3-5. Collision constraints

The algorithm for one time step of guide strand simulation:

- Add external forces
- Integrate using Verlet integration
- Repeat for number of iterations
    - Apply distance constraints
    - Apply angular constraints
    - Apply collision constraints

Since a given vertex can be constrained by the vertex before it and the one after it. We cannot apply all constraints in parallel. Instead we split the hair particles into two groups and alternate the constraints Figure 3-6 (left, right).

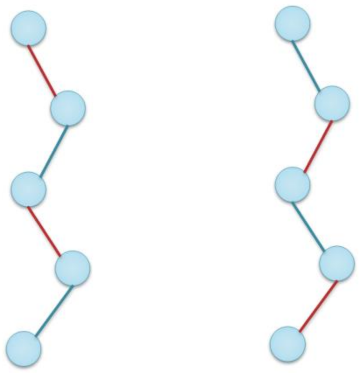

Figure 3-6. Constraints are applied in two steps

3.2.2 Hair Interpolation

After the simulation of guide strands, we pass our data from the compute pipeline to rendering pipeline. Then we utilize the Tessellation stage to interpolate the guide strands to create new hair strands for our dense hair model.

3.2.2.1 Interpolation methods

We use the Single strand based interpolation to create more hair strands. Using this interpolation each interpolated strand is created by offsetting it from a single guide strand along that strand's local coordinate frame. Each interpolated strand is assigned a random vector to offset it along the x and z coordinate axes.

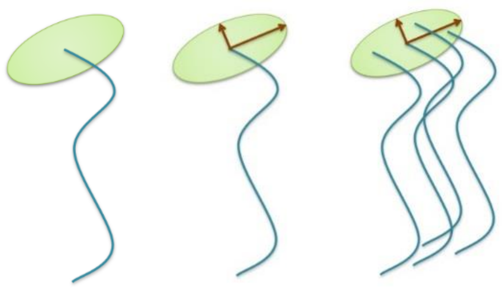

Figure 3-7. Single strand based interpolation

3.2.2.2 Hair tessellation

Tessellated hair strands are generated using Tessellation stage in the graphics rendering pipeline. As we know, there are three components in Tessellation stage: Hull Shader, Tessellator and Domain Shader. We are going to use the isoline tessellation domain to create new hair strands. In this mode the hardware Tessellator creates a number of isolines (connected line segments) with multiple line segments per line. We can specify the number of isolines and the number of segments per isoline in the Hull Shader. The actual positions and attributes of each tessellated vertex are evaluated using the Domain Shader. The output from the Tessellation stage is a set of line segments which can be rendered directly as lines, or they can be rendered as triangles by expanding them to camera facing quads using the geometry shader.

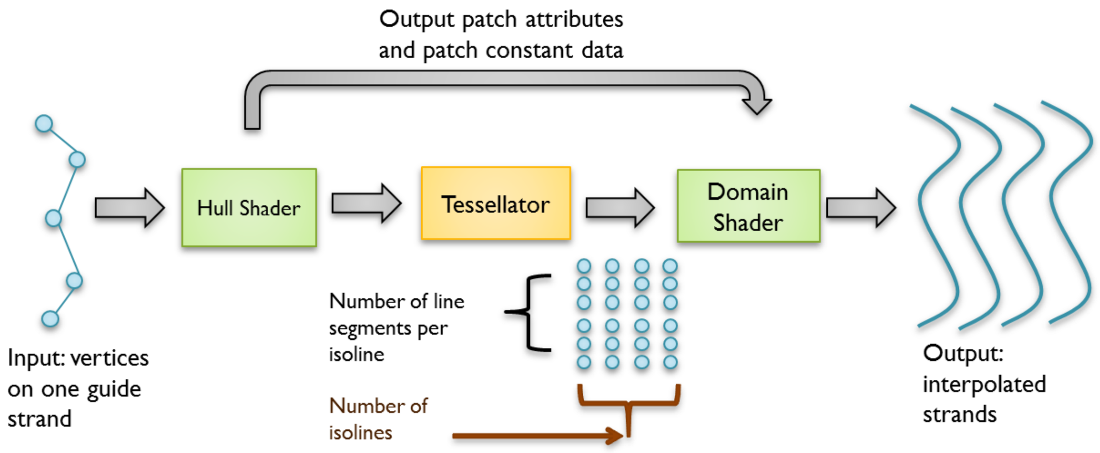
Output patch attributes and patch constant data
Hull Shader
Tessellator
Domain Shader
Input: vertices on one guide strand
Number of line segments per isoline
Number of isolines
Output: interpolated strands

Figure 3-8. An overview of a pass to interpolate guide strands to create more hair strands using the Tessellation stage

Our input is a patch of vertices on one single guide strand. We compute two tessellation factors which are the number of isolines and the number of line segments per isoline in the Hull Shader. The number of isolines is the number of new hair strands we want to create. The number of line segments will define how smooth our strands are going to be. These two factors are passed to the Tessellator. The Hull Shader also calculates any per patch data which might be needed by the Domain Shader. The Tessellator generates the topology requested by the Hull shader. Finally the strands are tessellated and interpolated in the Domain Shader.

The Hull Shader is consisted of two parts, the main shader and the patch constant function. The main shader operates once on each input control point, and the patch constant function which is invoked once per patch. In our implementation we are loading control point data for a patch from a buffer, so the input to the hull shader is a dummy patch consisting of a single control point. Since we have only one input control point, we use the patch constant function for all of the computation and our main shader is null. The patch constant function calculates the tessellation factors and other data for each patch.

The Domain Shader is invoked for each final vertex that is created. The input to the Domain Shader is what we have output from the Hull Shader, the patch constant data and the per patch control point data. We also get the id of the patch that we are operating on. Using these values we can figure out which vertex and which strand we are operating on. Then we can get our positions of control points from the buffer using this information. We also get a number of system generated input, including the parametric uv coordinates of the current vertex in the tessellated patch. u is corresponding to number of isolines we are creating. v is corresponding to the number of line segments per isoline. We can use v for parametric value together with our control points to generate a C1 continuous Catmull-Rom Spline to connect the vertices along one single strand. By defining consecutive four control vertices, we can get smoother spline instead of the separated chains. The larger the number of line segments per isoline was defined in Hull Shader, the smoother the spline will be. Given the control points P_0, P_1, P_2 and P_3 and the value t, the position of the desired segment can be calculated as:

$$q(t) = 0.5 * (1.0f, t, t^2, t^3) * \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} * \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

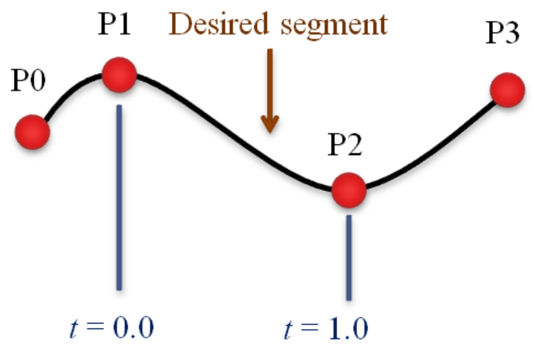

Figure 3-9. C1 continuous Catmull-Rom Spline

3.2.3 Hair Rendering

Our outputs are isolines which are not good enough for the rendering process. Due to the thinness of lines, rendering hair strands directly using lines could cause aliasing problem. Our solution is to expand the lines into camera facing quads using the Geometry Shader.

Then our hair strands are rendered using a commonly used hair shading model by by Kajiya and Kay [21]. The Kajiya and Kay model is the anisotropic strand lighting model which the hair strand is modeled as an infinitely thin cylinder. And the standard Phong lighting computations are adapted to use the hair tangent instead of a surface normal. We compute the final color of using the parameters including the hair tangent

(**T**), the view vector (**V**), the light vector (**L**), the half-angle vector (**H**), and the exponent s which specifies the sharpness of the highlight.

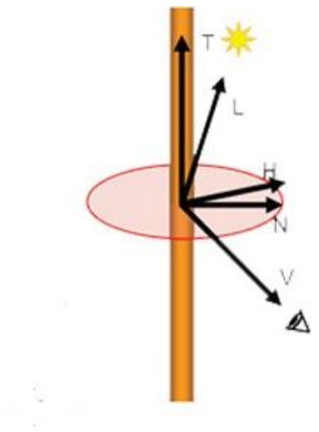

Figure 3-10. Kajiya and Kay model [21]

The diffuse and specular terms are computed as follows:

$$diffuse = sin(T, L) = \sqrt{1 - \cos(T, L)^2} = \sqrt{1 - (T \cdot L)^2}$$

$$specular = \sin(T, H)^s = \sqrt{1 - (T \cdot H)^2}^{s}$$

CHAPTER 4
IMPLEMENTATION RESULTS

We created the hair mesh model using modeling software 3ds Max. Then the hair mesh is exported and converted into guide strands. We simulate these guide strands to guide the motion of the full hair model. A dense hair model is created by interpolating the position of the remaining strands from the sparse set of guide strands. The final result is rendered by using the Kajiya & Kay model.

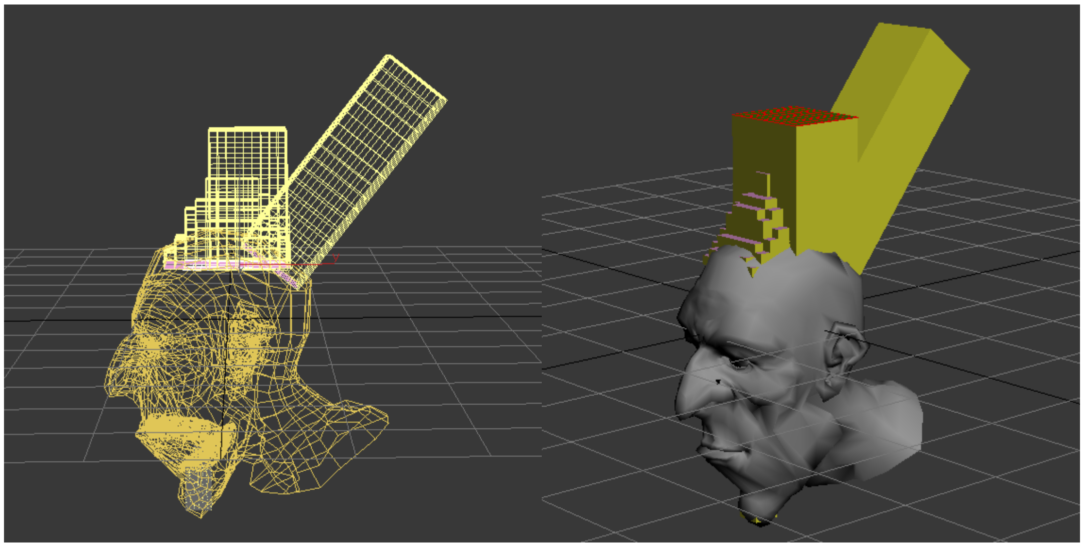

Figure 4-1. Hair mesh created in 3ds Max

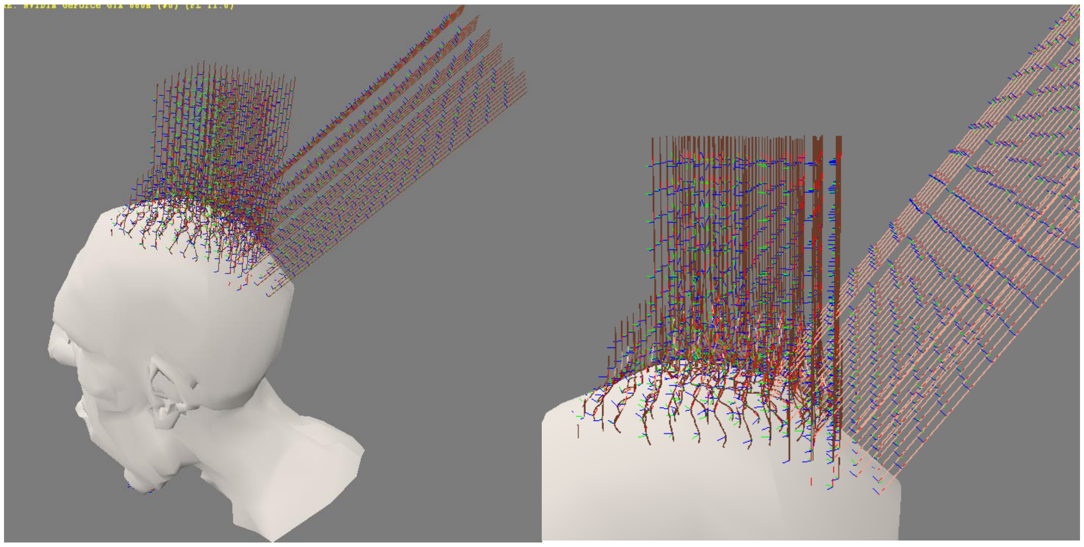

Figure 4-2. Hair mesh converted into guide hair strands

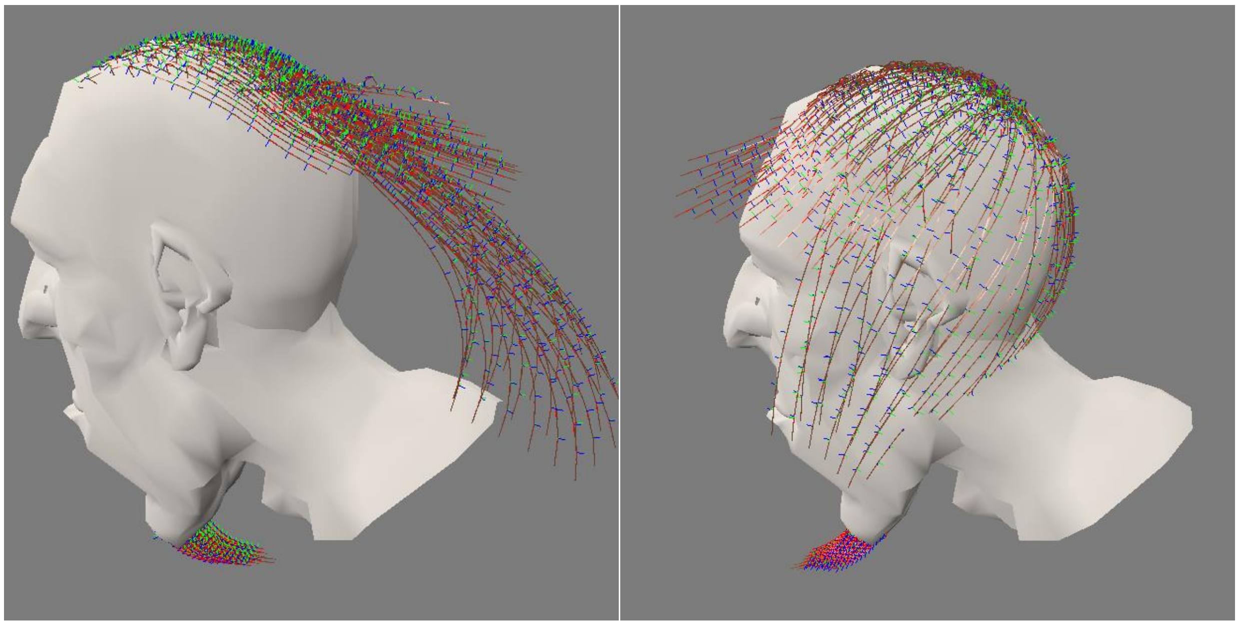

Figure 4-3. Real-time hair simulation with gravity; Green, red and blue axes represent local coordinates for each vertex

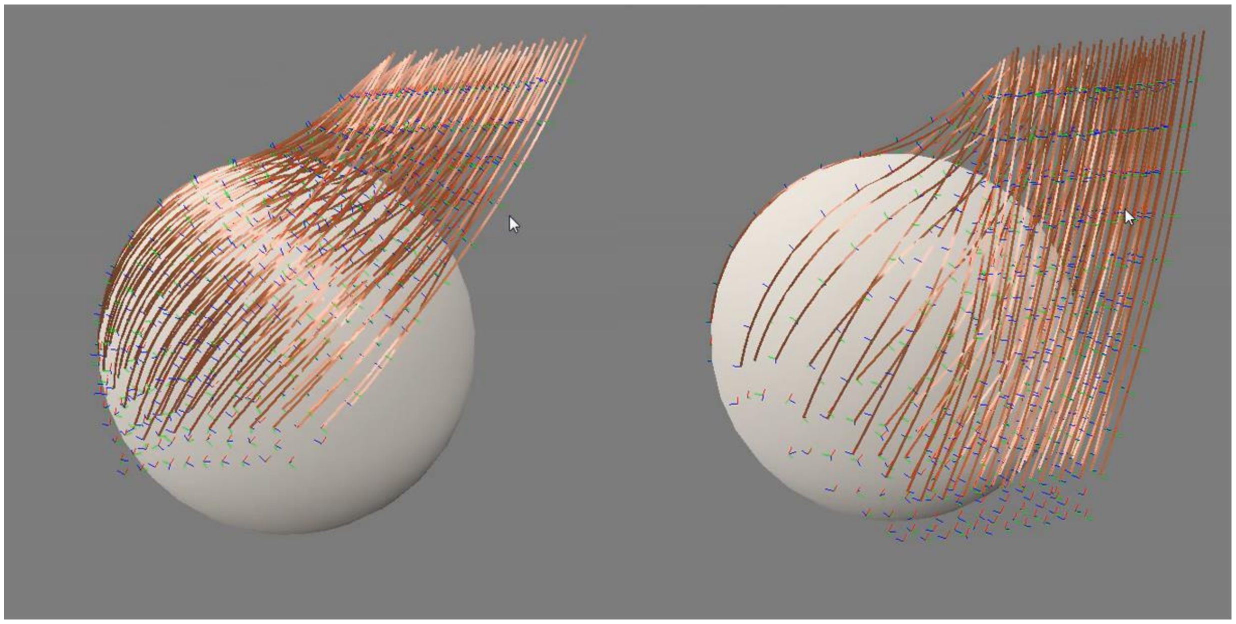

Figure 4-4. A simpler result shows the simulation with sphere collision

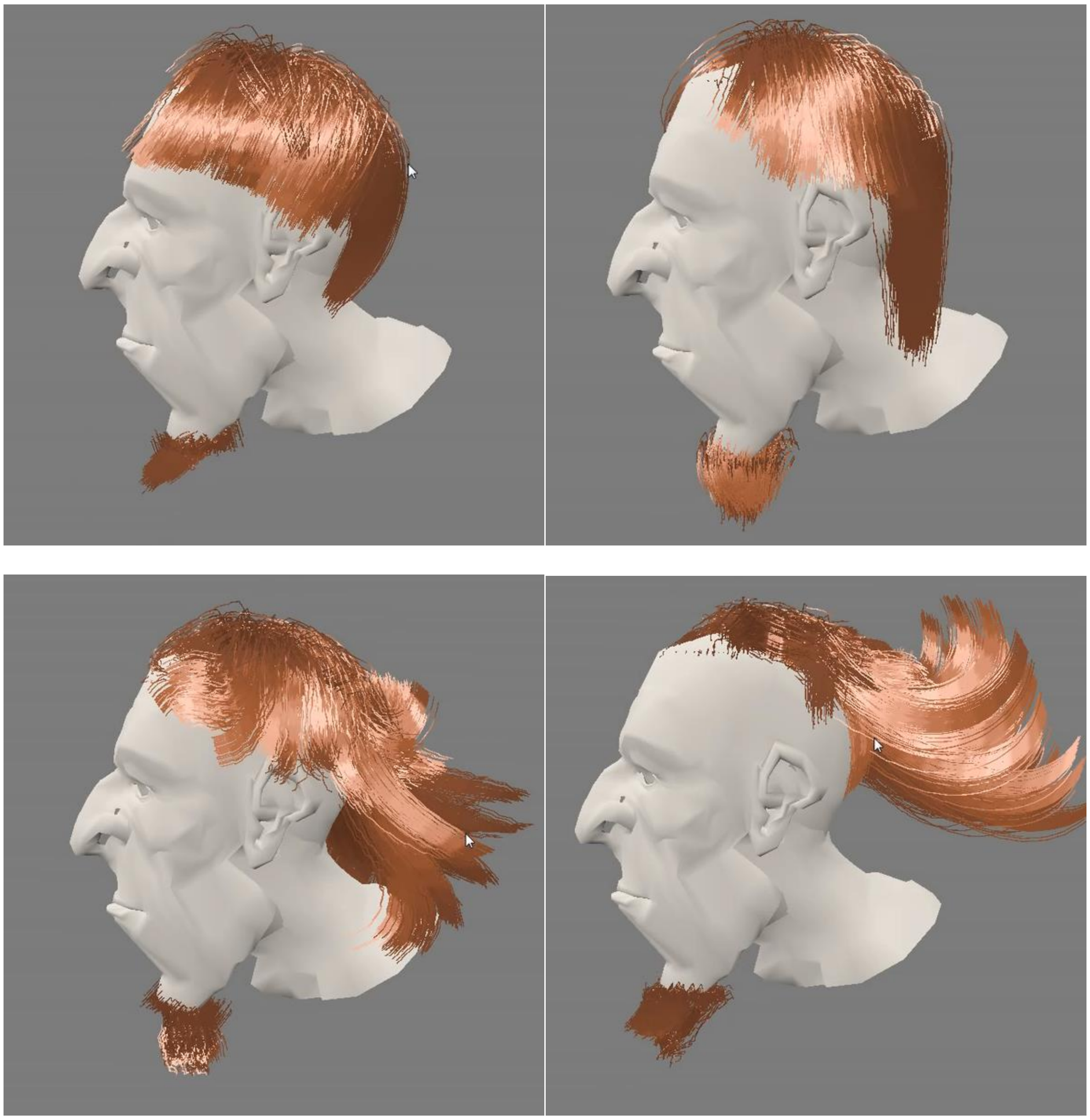

Figure 4-5. Strands interpolation and final results

The running speed of simulating 240 guide strands with more than 10 vertices per strand is close to 120 frames per second. Tested with GeForce GTX 660M.

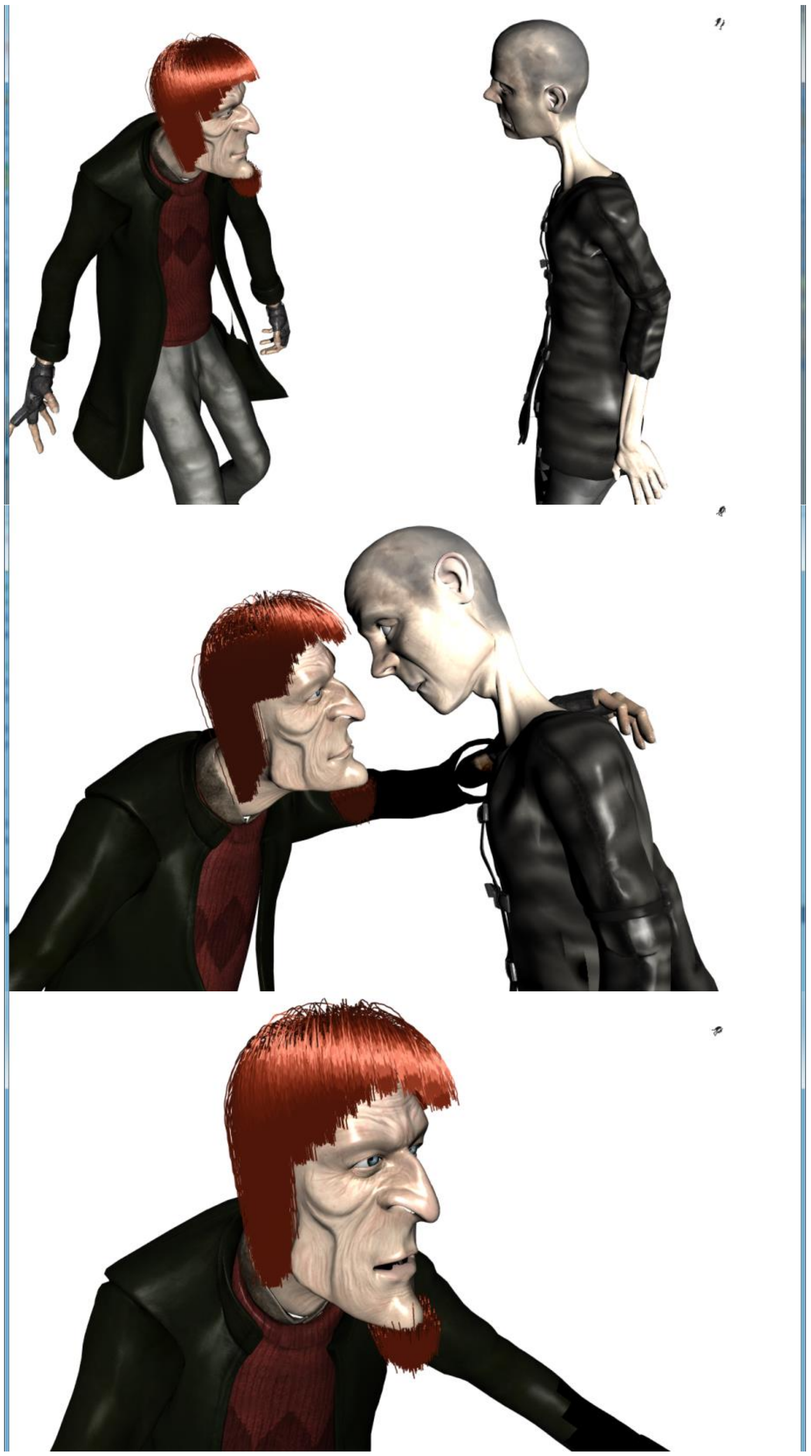

Figure 4-6. Final results after integrating hair modeling to the iPASS[54] project to render the open source movie “Elephant's Dream” in real-time

The running speed is close to 60 frames per second. Tested with GeForce GTX 660M.

CHAPTER 5
CONCLUSION

We presented a real-time hair modeling framework. The framework encompasses hairstyling, hair simulation, and hair rendering. The flexibility of our framework gives the users or artists fully control from designing the hairstyle to the final rendering of full hair model. Any kind of hair shape can be created in modeling software in the same way as any polygon mesh is created.

The usage of shader technology makes it very convenient for us to replace any component of our program in the future. For example, the simulation of Super-Helices could be exploited using GPU parallel computing. Then we can replace our Mass-Spring systems on the Compute Shader with the more accurate Super-Helices simulation model. More rendering techniques, for instance, self-shadowing can be implemented for our hair strands to generate more realistic results.

REFERENCES

[1] A. Daldegan, N. M. Thalmann, T. Kurihara, and D. Thalmann, “An integrated system for modeling, animating and rendering hair,” *Computer Graphics Forum*, vol. 12, no. 3, pp. 211–221, 1993.

[2] A. M. LeBlanc, R. Turner, and D. Thalmann, “Rendering hair using pixel blending and shadow buffers,” *The Journal of Visualization and Computer Animation*, vol. 2, no. 3, pp. 92–97, – 1991.

[3] Alter, Joseph Scott. "Hair generation and other natural phenomena with surface derived control volumes in computer graphics and animation." U.S. Patent 6,720,962, issued April 13, 2004.

[4] Anjyo, Ken-ichi, Yoshiaki Usami, and Tsuneya Kurihara. "A simple method for extracting the natural beauty of hair." In *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, pp. 111-120. ACM, 1992.

[5] C. Koh and Z. Huang, “A simple physics model to animate human hair modeled in 2D strips in real time,” in *Computer Animation and Simulation ’01*, Sept. 2001, pp. 127–138.

[6] Choe, Byoungwon, and Hyeong-Seok Ko. "A statistical wisp model and pseudophysical approaches for interactive hairstyle generation." *Visualization and Computer Graphics, IEEE Transactions on* 11, no. 2 (2005): 160-170.

[7] Daldegan, Agnes, Nadia Magnenat Thalmann, Tsuneya Kurihara, and Daniel Thalmann. "An integrated system for modeling, animating and rendering hair." In*Computer Graphics Forum*, vol. 12, no. 3, pp. 211-221. Blackwell Science Ltd, 1993.

[8] E. Plante, M.-P. Cani, and P. Poulin, “A layered wisp model for simulating interactions inside long hair.”

[9] E. Sugisaki, Y. Yu, K. Anjyo, and S. Morishima, “Simulation-based cartoon hair animation,” in *Proceedings of the 13th Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2005.

[10] F. Bertails, B. Audoly, B. Querleux, F. Leroy, J.-L. L´evˆeque, and M.-P. Cani, “Predicting natural hair shapes by solving the statics of flexible rods,” in *Eurographics (short papers)*, August 2005.

[11] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, and J.-L. L´evˆeque, “Super-helices for predicting the dynamics of natural hair,” in *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, August 2006.

[12] F. Bertails, T.-Y. Kim, M.-P. Cani, and U. Neumann, “Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion,” in *ACM SIGGRAPH Symposium on Computer Animation*, July 2003, pp. 207–213.

[13] Fu, Hongbo, Yichen Wei, Chiew-Lan Tai, and Long Quan. "Sketching hairstyles." In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pp. 31-36. ACM, 2007.

[14] Gee, K. "DirectX 11 Tessellation". In *Microsoft GameFest,* 2008.

[15] Grabli, Stephane, François X. Sillion, Stephen R. Marschner, and Jerome E. Lengyel. "Image-based hair capture by inverse lighting." In *Proceedings of Graphics Interface (GI)*, pp. 51-58. 2002.

[16] Gupta, Rajeev, Melanie Montagnol, Pascal Volino, and Nadia Magnenat-Thalmann. "Optimized framework for real time hair simulation." In *Advances in Computer Graphics*, pp. 702-710. Springer Berlin Heidelberg, 2006.

[17] Gupta, Rajeev, Melanie Montagnol, Pascal Volino, and Nadia Magnenat-Thalmann. "Optimized framework for real time hair simulation." In *Advances in Computer Graphics*, pp. 702-710. Springer Berlin Heidelberg, 2006.

[18] H. D. Taskiran and U. Gudukbay, “Physically-based simulation of hair strips in real-time,” in *Proceedings of the 13th Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2005.

[19] H. Nguyen and W. Donnelly, \Hair Animation and Rendering in the Nalu Demo," in *GPU Gems 2* (M. Pharr and R. Fernando, eds.), ch. 23, Addison Wesley Professional, Mar. 2005.

[20] Hadap, Sunil, and Nadia Magnenat-Thalmann. "Interactive hair styler based on fluid flow." In *Computer Animation and Simulation 2000*, pp. 87-99. Springer Vienna, 2000.

[21] J. Kajiya and T. Kay, “Rendering fur with three dimensional textures,” in *Proceedings of ACM SIGGRAPH 89*, ser. Computer Graphics Proceedings, Annual Conference Series, 1989, pp. 271–280.

[22] J. T. Chang, J. Jin, and Y. Yu, “A practical model for hair mutual interactions,” in *ACM SIGGRAPH Symposium on Computer Animation*, July 2002, pp. 73–80.

[23] K. Anjyo, Y. Usami, and T. Kurihara, “A simple method for extracting the natural beauty of hair,” in *Proceedings of ACM SIGGRAPH 1992*, ser. Computer Graphics Proceedings, Annual Conference Series, August 1992, pp. 111–120.

[24] Kim, T-Y., and Ulrich Neumann. "A thin shell volume for modeling human hair." In *Computer Animation 2000. Proceedings*, pp. 104-111. IEEE, 2000.

[25] Koh, Chuan Koon, and Zhiyong Huang. "A simple physics model to animate human hair modeled in 2D strips in real time." In *Computer Animation and Simulation 2001*, pp. 127-138. Springer Vienna, 2001.

[26] Kong, Waiming, and Masayuki Nakajima. "Generation of 3d hair model from multiple pictures." *The Journal of the Institute of Image Information and Television Engineers* 52, no. 9 (1998): 1351-1356.

[27] Lee, Doo-Won, and Hyeong-Seok Ko. "Natural hairstyle modeling and animation." *Graphical Models* 63, no. 2 (2001): 67-85.

[28] Liang, Wenqi, and Zhiyong Huang. "An Enhanced Framework for Real-Time Hair Animation." In *Pacific Conference on Computer Graphics and Applications*, pp. 467-471. 2003.

[29] Malik, Shahzad. "A sketching interface for modeling and editing hairstyles." In*Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pp. 185-194. The Eurographics Association, 2005.

[30] Mao, Xiaoyang, Shiho Isobe, Ken Anjyo, and Atsumi Imamiya. "Sketchy hairstyles." In *Computer Graphics International 2005*, pp. 142-147. IEEE, 2005.

[31] Moon, Jonathan T., and Stephen R. Marschner. "Simulating multiple scattering in hair using a photon mapping approach." In *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 1067-1074. ACM, 2006.

[32] N. Magnenat-Thalmann and S. Hadap, “State of the art in hair simulation,” In *International Workshop on Human Modeling and Animation*, ser. Korea Computer Graphics Society, June 2000, pp. 3–9.

[33] Noble, Paul, and Wen Tang. "Modelling and animating cartoon hair with nurbs surfaces." In *Computer Graphics International, 2004. Proceedings*, pp. 60-67. IEEE, 2004.

[34] P. Volino and N. Magnenat-Thalmann, “Animating complex hairstyles in real-time,” in *ACM Symposium on Virtual Reality Software and Technology*, 2004.

[35] Pai, Dinesh K. "Strands: Interactive simulation of thin solids using cosserat models." In *Computer Graphics Forum*, vol. 21, no. 3, pp. 347-352. Blackwell Publishing, Inc, 2002.

[36] Paris, Sylvain, Hector M. Briceño, and François X. Sillion. "Capture of hair geometry from multiple images." In *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 712-719. ACM, 2004.

[37] Paris, Sylvain, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. "Hair photobooth: geometric and photometric acquisition of real hairstyles." *ACM Trans. Graph* 27, no. 3 (2008): 30.

[38] R. Featherstone, *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[39] R. Rosenblum, W. Carlson, and E. Tripp, “Simulating the structure and dynamics of human hair: Modeling, rendering, and animation,” *The Journal of Visualization and Computer Animation*, vol. 2, no. 4, pp. 141–148, 1991.

[40] S. Hadap and N. Magnenat-Thalmann, “Modeling dynamic hair as a continuum,” *Computer Graphics Forum*, vol. 20, no. 3, pp. 329–338, 2001, proceedings of Eurographics’01.

[41] S. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan,“Light scattering from human hair fibers,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 780–791, July 2003, proceedings of ACM SIGGRAPH 2003.

[42] S. Tariq and L. Bavoil, "Real time hair simulation and rendering on the gpu," in *ACM SIGGRAPH 2008 talks*, SIGGRAPH '08, (New York, NY, USA), pp. 37:1- 37:1, ACM, 2008.

[43] T. Kurihara, K. Anjyo, and D. Thalmann, “Hair animation with collision detection,” in *Proceedings of Computer Animation’93*. Springer, 1993, pp. 128–138.

[44] T.-Y. Kim and U. Neumann, “A thin shell volume for modeling human hair,” in *Computer Animation 2000*, ser. IEEE Computer Society, 2000, pp. 121–128.

[45] W. Liang and Z. Huang, “An enhanced framework for real-time hair animation,” in *Pacific Graphics Conference on Computer Graphics and Applications*, October 2003.

[46] Wang, Lvdi, Yizhou Yu, Kun Zhou, and Baining Guo. "Example-based hair geometry synthesis." *ACM Transactions on Graphics (TOG)* 28, no. 3 (2009): 56.

[47] Ward, Kelly, Nico Galoppo, and Ming Lin. "Interactive virtual hair salon."*Presence: Teleoperators and Virtual Environments* 16, no. 3 (2007): 237-251.

[48] Wei, Yichen, Eyal Ofek, Long Quan, and Heung-Yeung Shum. "Modeling hair from multiple views." In *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 816-820. ACM, 2005.

[49] Wither, Jamie, Florence Bertails, and M-P. Cani. "Realistic hair from a sketch." In *Shape Modeling and Applications, 2007. SMI'07. IEEE International Conference on*, pp. 33-42. IEEE, 2007.

[50] X. D. Yang, Z. Xu, T. Wang, and J. Yang, “The cluster hair model,” *Graphics Models and Image Processing*, vol. 62, no. 2, pp. 85–103, Mar. 2000.

[51] Y. Bando, B.-Y. Chen, and T. Nishita, “Animating hair with loosely connected particles,” *Computer Graphics Forum*, vol. 22, no. 3, pp. 411– 418, 2003, proceedings of Eurographics’03.

[52] Y. Guang and H. Zhiyong, “A method of human short hair modeling and real time animation,” in *Pacific Graphics*, Sept. 2002.

[53] Y. Watanabe and Y. Suenaga, “A trigonal prism-based method for hair image generation,” *IEEE Computer Graphics and Applications*, vol. 12, no. 1, pp. 47–53, Jan 1992.

[54] Yeo, Young In, Lihan Bin, and Jörg Peters. "Efficient pixel-accurate rendering of curved surfaces." In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 165-174. ACM, 2012.

[55] Yu, Yizhou. "Modeling realistic virtual hairstyles." In *Computer Graphics and Applications, 2001. Proceedings. Ninth Pacific Conference on*, pp. 295-304. IEEE, 2001.

[56] Yukesl, D. Pai, “Strands: Interactive simulation of thin solids using cosserat models,” Computer Graphics Forum, vol. 21, no. 3, pp. 347–352, 2002, proceedings of Eurographics’02.

[57] Yuksel, Cem, Ergun Akleman, and John Keyser. "Practical global illumination for hair rendering." In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pp. 415-418. IEEE, 2007.

[58] Yuksel, Cem, Scott Schaefer, and John Keyser. "Hair meshes." In *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 166. ACM, 2009.

[59] Zinke, Arno, Cem Yuksel, Andreas Weber, and John Keyser. "Dual scattering approximation for fast multiple scattering in hair." In *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, p. 32. ACM, 2008.

BIOGRAPHICAL SKETCH

Yang Chen was born in Xianyou, China. He received his Bachelor of Engineering degree (2012) from University of Macau. His work during his master’s degree focused on computer graphics, computer simulation and GPU techniques.