國立交通大學

多媒體工程研究所

碩 士 論 文

漫 畫 風 格 髮 型 即 時 成 像 之 研 究

Real-time Stylized Line-based Rendering of Animated Hairs

研 究 生：張又升

指導教授：施仁忠　教授

魏德樂　教授

中 華 民 國 一 百 零 一 年 六 月

漫畫風格髮型即時成像之研究

Real-time Stylized Line-based Rendering of Animated Hairs

研　究　生：張又升　　　　Student：Yu-Sheng Chang

指導教授：施仁忠　　　　Advisor：Prof. Zen-Chung Shih

魏德樂　　　　Prof. Der-Lor Way

國　立　交　通　大　學

多　媒　體　工　程　研　究　所

碩　士　論　文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Multimedia Engineering

June 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年六月

漫畫風格髮型即時成像之研究

研究生: 張又升　　　　　　　　　　　指導教授: 施仁忠教授

魏德樂教授

國立交通大學多媒體工程研究所

摘　　　　要

在漫畫與卡通動畫中，頭髮對於畫家來說，是個展現角色的個人特質、情感、以及特色風格的一項重要工具。本論文提出一個動漫畫風格髮絲動畫的即時成像技術。我們首先將動態髮絲上的質點做為系統的輸入。而後我們利用非擬真顯像領域中的 stylized line-based rendering 的技術，套用於動態髮絲上來模擬動漫畫家的繪圖筆觸風格。我們所產生的筆觸效果包括影線、卡通著色法、強光羽狀筆觸、以及其他動漫畫特效。筆觸的模擬與產生，以及最後的成像皆於 GPU 上執行並完成，利用 GPU 的硬體優勢來有效提升執行效率。

Real-time Stylized Line-based Rendering of Animated Hairs

Student: Yi-Sheng Chang Advisor: Prof. Zen-Chung Shih

Prof. Der-Lor Way

Institute of Multimedia Engineering

National Chiao-Tung University

ABSTRACT

In comics and cartoons animation, hair is a strong vehicle for artists to express the personality, emotion, and style of the character. In this thesis, we present a real-time rendering technique for the hair animation of comic and cartoon styles hair. We first take a set of dynamic particles to generate the animated hair strands. Then we use a stylized line-based rendering technique performing on the generated hair strands to generate the stylized hair strokes that imitate the comic drawings. The stroke effects we generate include silhouettes, cel-shading, highlights, and other comic effects. Strokes generation and rendering are performed on GPU and utilize the advance of graphics hardware to improve the performance.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, Pro. Zen-Chung Shih and Prof. Der-Lor Way for his guidance and patience. Without their encouragement, I would not complete this thesis. Thanks also to all the members in Computer Graphics and Virtual Reality Laboratory for their reinforcement and suggestion. Thanks for those people who have supported me during these days. Finally, I want to dedicate the achievement of this work to my family and the natural beauty around daily life.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

The Comics, Ninth Art, get more and more important in the entertainment industry like movies, animations, and games. The comic artists use the particular brush strokes and contours to delicately express the rich contents and visual effects. Hair is an important visual feature of characters in cartoon animations and comics. The artists not only draw hairs to express the personality, emotion, and style of the character, but also use specific brushworks to express the rich dynamics of hairs.

Hair simulation is one of the major topics in Computer Graphics, including hair modeling, dynamic simulation, hair rendering …etc., but most of the papers focus on realistic hair simulation. Here we introduce a particle-based dynamic simulation method, and combine with the stroke-based simulation algorithm which is famous in Non-photorealistic rendering (NPR) to generate the comic-style rendering of dynamic hair animation.

Non-photorealistic rendering (NPR) algorithm, unlike traditional rendering to imitate reality, create a variety appearance of stylistic drawings like oil painting, watercolor, and comics. One of the most popular methods is the stroke-based rendering, which generates and mimics the artistic strokes to produce the final painting. Although many methods can successfully create desired effect or even perform well on many industry applications, there is still some details could be improved. We focus on the hairs of the character and take comic and cartoon styles as examples, improve traditional NPR method to create more delicate expression like artist's drawing.

Our contributions are summarized as follow:

1. The proposed algorithm combining stylized line-based rendering method with dynamic hair strands to generate the hair strokes with following comic-style effects: silhouettes, cel-shading, highlights, and thin fine hairs.
2. A simple hair stroke model based on particle-based dynamic simulation to generate the hair brush stroke paths.
3. A real-time implementation based on GPU, which is allowed to perform on many applications in the entertainment industry like movies, animations, and games.
4. A re-ordering mechanism to deal with the depth and stroke rendering orders, which make a good compromise between performance and temporal coherence.

1.2 System Overview

Our system generates the comic-style hairs animation by the following steps: animated hair strands generation, stroke path generation, hair strokes generation, and rendering. Figure 1.1 shows the flow of our proposed system.

First of all, we use particle-based dynamic system based on Chang et al. [6] to generate enough animated hairs strands. These hair strands will be used as a guide to generate the stroke paths in screen space that will later be used in stylized line-based rendering method.

Then the system will generate the brush strokes applied on the stroke paths to imitate the comic-style hairs. We use the following shaders to generate different brush stroke effects: base shader, silhouette shader, cel-shading shader, highlight shader, and thin fine hairs shader. Base shader consists of base information like size of strokes and others. Silhouette shader generates some feature lines like boundaries and occlusive contours. Cel-shading shader generates the cel-shaded stroke effects. The highlight shader generates the highlight effect. Thin fine hairs shader produces the thin fine hairs effects which are common in comics. We will discuss the details of each shader later in Chapter 5, and the user can edit some parameters of each shader or choose which shader would be activated to produce the desired effects.

Finally, the system will combine all of the activated shaders and render in specific order by our re-ordering mechanism with temporal coherence in real-time.

1.3 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 presents the related works of NPR and some previous methods on simulating carton or comic hairs. Chapter 3 describes the effects of hairs in comics and cartoon, and the rules of stylized line-based rendering, later with dynamic hair simulation system we used. The hair and stroke model are described in Chapter 4, and then we discuss the stroke simulation of each effect in detail in Chapter 5. Finally, the results are shown in Chapter 6, and in Chapter 7 we conclude this research.

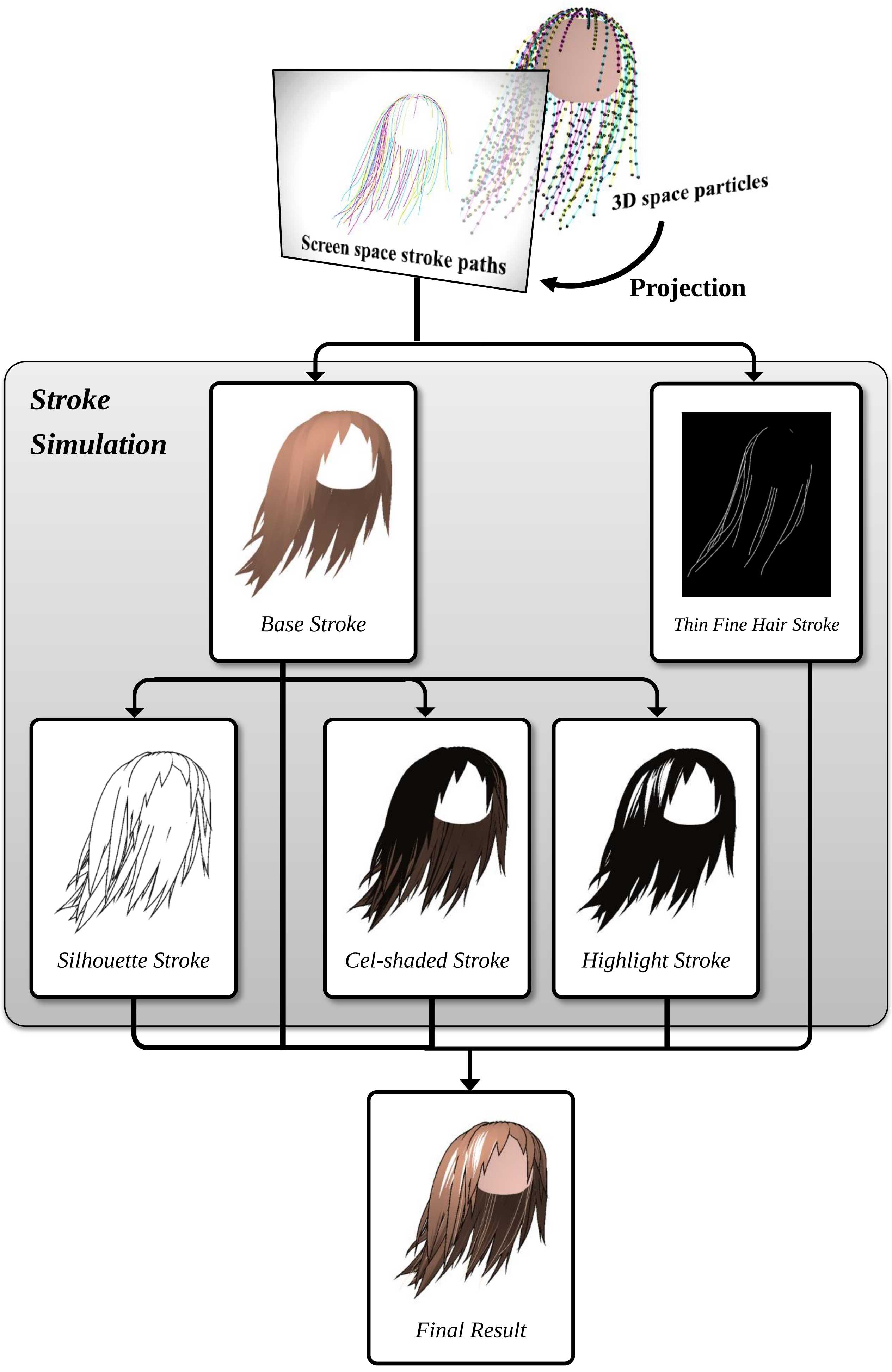


Figure 1.1: System overview

Chapter 2
Related Works

In this chapter, we review some previous works. First we describe the Non-Photorealistic Rendering algorithms for art and some issues should be concerned when rendering. Then we discuss the previous hair simulation methods on imitating the comic or cartoon effects.

2.1 Non-Photorealistic Rendering

Non-Photorealistic Rendering (NPR) is interesting because of its expressive power for generating artistic drawing and paintings. Researches in painterly rendering have demonstrated that certain artistic styles can be mimicked. One of the most popular methods is stroke-based rendering, which generates and mimics the artistic strokes and places at specific positions to produce the final painting. Here we use the stylized line-based rendering, which takes lines as the stroke paths and apply strokes on them.

Determining where or which lines should be taken as the stroke paths is also an interesting topic. Many of the stylized line-based rendering relies on automatically finding the feature lines in 3D geometry, like suggestive contours [8], ridge-valley lines [19], apparent ridges [13], demarcating curves [17], and Laplacian lines [29]. These feature lines were found on 3D surfaces by computing either the 2nd order (such a suggestive contours) or 3rd order (such as ridge-valley lines, apparent ridges, and Laplacian lines) derivatives of certain properties like depth, curvature, or diffuse illumination. Artists often use boundary lines and silhouettes to depict the hairs. But our method is different from them since we use particle dynamic system to generate hairs as input, which lead to no 3D mesh information. Our method to generate the silhouettes strokes without mesh information will discuss in Chapter 5.

Another issue in NPR is the temporal coherence in animation. It should be taken into account to avoid some unpleasant results like popping, sliding or any other discontinuous artifacts. Bénard et al. [2] surveyed recent researches on temporal coherence in NPR animation and proposed that the NPR animation should satisfy flatness, motion coherence, and temporal continuity properties. Stroke-based methods are often suffered from temporal continuity, due to some discontinuous information between frames. Bénard give the basis to comment whether an NPR animation is good or not, we will discuss our result with these properties carefully in Chapter 6.

2.2 Hair Simulation

Hair simulation is essential to computer graphics, most of the papers focus on realistic representation of hair for various applications. Ward et al. [24] surveyed the recent hair simulation techniques on several major topics. There are lots of methods to simulate the dynamic of the hairs, either by particles or polygon meshes. When rendering stage, most of the methods generate the hair strands from their primitive data. Therefore, we use the dynamic hair strands as input to simulation the hair strokes. Any dynamic system that generates the hair strands can be our input. In this thesis, the dynamic simulation of hair is based on Chang et al. [6] to generate the hair stands. Although most papers focus on realistic hair simulation, there are still a few researches devoted to cartoon representation of hair.

Noble et al. [18] proposed a method to generate NURBS volumes from hair strands then rendering with cel-shading. It produces the clumpy look of cartoon hair, but lack of stroke effect and other special effects like highlights that artists often use.

Cote et al. [7] presented a procedure that takes image as input then drawing hatches along the hair orientation and generating highlights with ink stain. Their method mimics the artists' strokes well especially on highlight regions. It generates the hair animation by interpolating various keyframes which are set by user. This approach is different from generating and rendering from 3D hair strands in real-time.

Shin et al. [23] proposed a stylized cartoon hair render, which is similar to our approach. They took dynamic particles as input to produce the hair strands. Although they offer different shaders, but the results still lack of stroke effects and are different from the way when artists draw the hair. Our methods also take 3D dynamic particles as input and will run in real-time with different stroke effects, which give the rich applications in the entertainment industry.

Chapter 3 Background

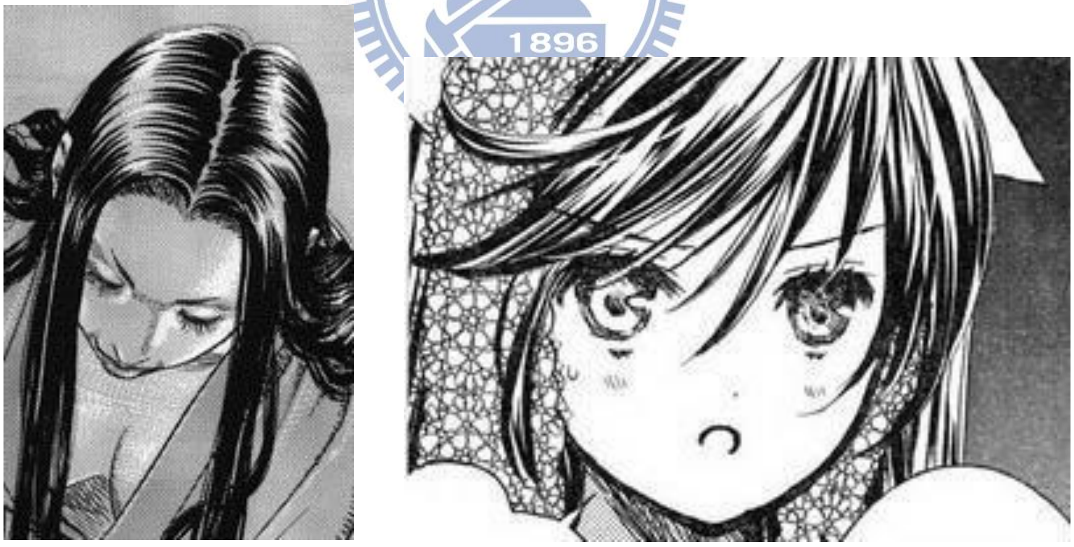
3.1 Hairs in Comics and Cartoon Animation

Comic is a form of static 2-dimensional illustrated visual art, and cartoon can be viewed as its animation form. In the section, we briefly introduce how artists draw the hairs in comics and cartoon.

Figure 3.1 shows different NPR hair styles in comics. Figure 3.2 display some detail effects by zooming in parts of the hair. From Figures 3.1 and 3.2, we can find that different artists might use different effects to express the hairs. For example, comics in Figure 3.1 are all black and white and the white strokes are the highlights. Figure 3.2 (a) uses silhouettes, cel-shading, and highlights, while Figure 3.2(b) has no silhouettes but some thin fine hairs. We analyze and organize these effects in different shaders, and then the users could create the desired effects by choosing which effect shaders to be activated. The effects we implement in our system are as follows: base stroke (the basic size and color), cel-shaded stroke (the cel-shading effect with lighter or darker color), silhouette stroke, highlight stroke, and thin fine hair stroke. Figure 3.3 gives the examples of hair animation in cartoons. We capture a few frames of the animation. We can find the effects mentioned above in these animation examples.

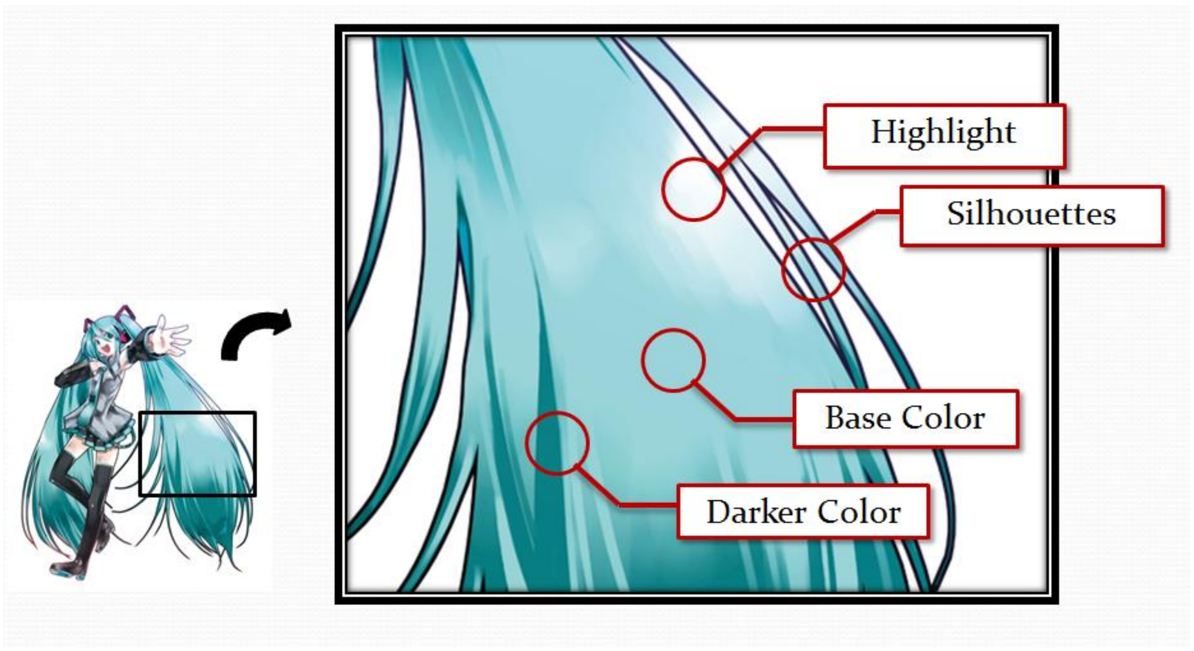
The next issue is how the artist draws these hair effects. From the example figures, the comic artist usually draws a clump of hairs instead of each single hair separately. Thus, unlike realistic hair simulation, we just need few hair strands to construct a clumpy look of hairs, which can reduce the computation time in simulating both the dynamic particle system and the brush strokes. We use the brush models of Weng [25] and Ho [12] to imitate the artists' brushwork.

The order of drawing each stroke effect is also important. Artists often first draw the outlines and silhouettes of hairs, and apply the basic color. Next they apply cel-shading color effects and highlights. Finally, some thin fine hairs are drawn to produce more dynamic effects if needed. When rendering, the stroke rendering order should combine with the depth information to generate the correct results.

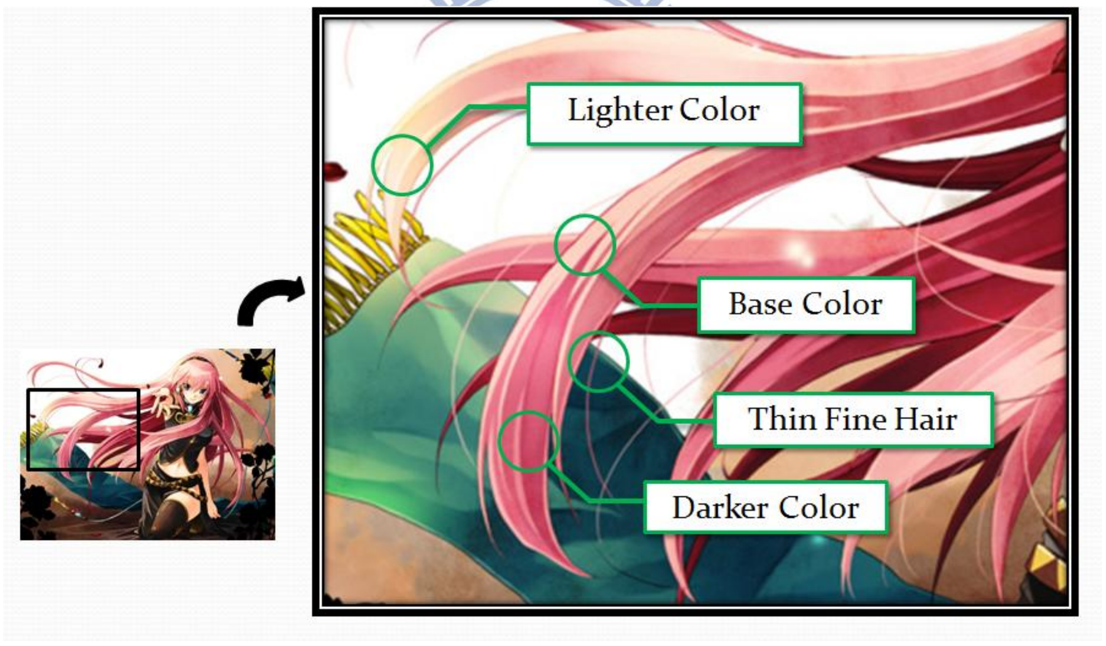


(a) Artwork by Takehiko Inoue (b) Artwork by PEACH-PIT

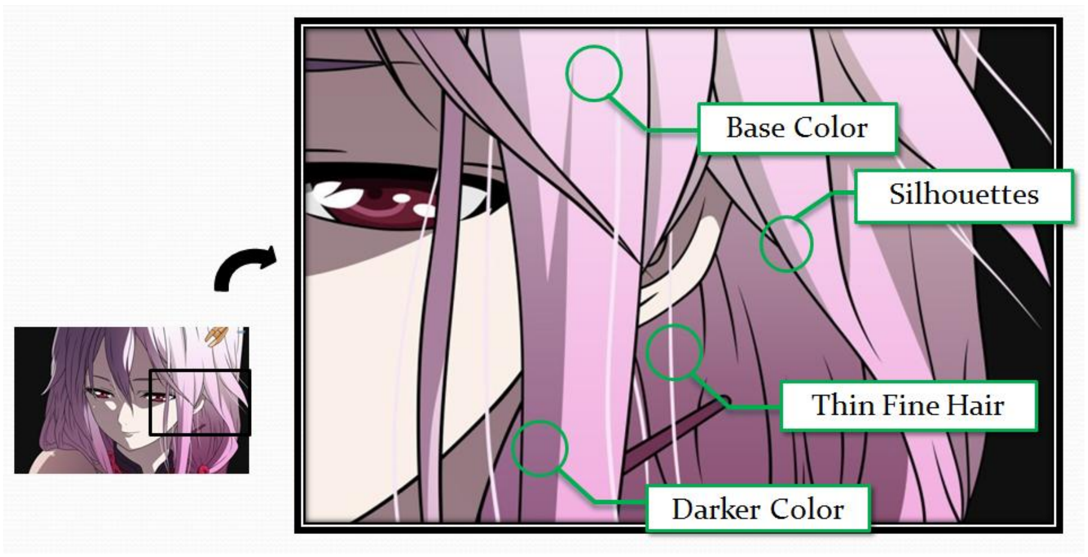
Figure 3.1: Black and white comic examples



(a) Hatsune Miku from Cryton Future Media

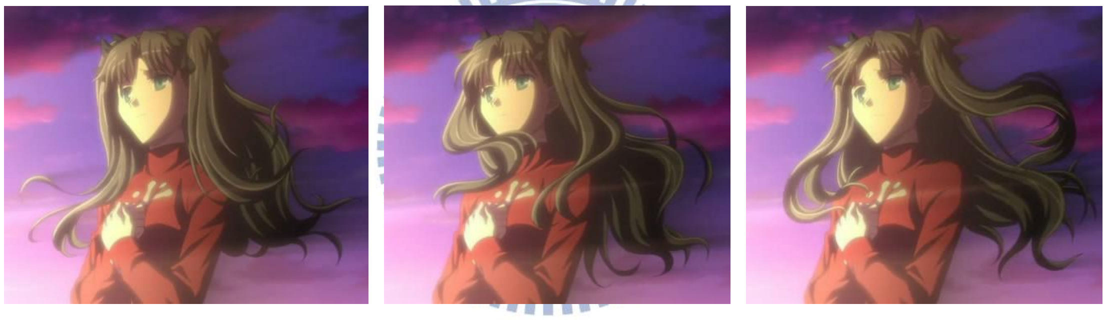


(b) Megurine Luka from Cryton Future Media

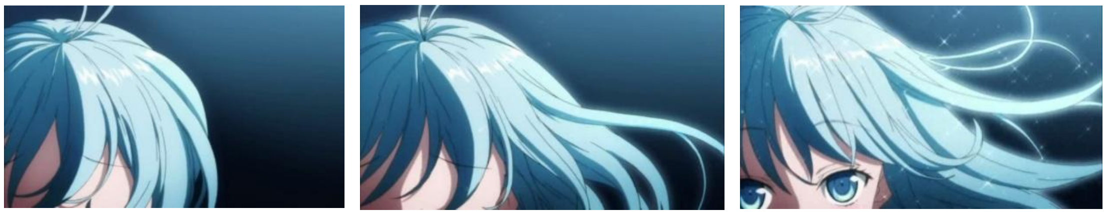


(c) Character form animation “Guilty Crown”

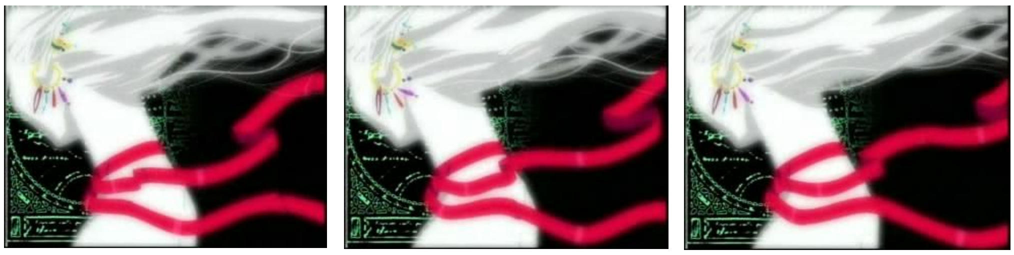
Figure 3.2: Color comic examples



(a) Animation “Fate/Stay Night - UNLIMITED BLADE WORKS (2010)”



(b) Animation “Denpa Onna to Seishun Otoko (2011)”



(c) Animation “Record of Lodoss War OVA (1990)”

Figure 3.3: Cartoon animation examples

3.2 Stylized Line-based Rendering

Computer-aided line-based rendering takes lines as the stroke paths and apply brush strokes on it. These lines are often the object's feature lines such as boundaries, silhouettes, ridges, and valleys. By applying different styles of brush strokes to them, we can obtain rich expressions of stylized line drawings. But there are two major challenges: one is how to extract the feature lines automatically, the other is how to render them in temporal coherent manner.

Our system uses stylized line-based rendering technique which is quite different from the traditional purpose. We use this technique to generate the hair strokes, which are not only the line drawings of the hairs, but the whole representation of the final strokes with several effects like cel-shading and highlights. Thus traditional feature lines of object are no longer the input of the stylized line-based rendering technique we use. But the final representation still needs these feature lines as mentioned before, we propose a different method to generate the feature lines in the hair stroke, details will be discussed in Chapter 5.

The second challenge is to ensure temporal coherence when rendering. Take the idea of Bénard et al. [2] that temporal coherence of NPR animation should ensure flatness, motion coherence, and temporal continuity properties. Most stroke-based approaches of stylization first project feature points to screen space and apply brush strokes, thus the flatness is satisfied due to screen-space rendering. Motion coherence is usually satisfied for line-based rendering inherently because the strokes are stuck on the moving feature lines. But these methods are suffered from temporal continuity due to the parameterization process might change when any discontinuity occurs, which lead to popping or sliding artifacts. Our line-based rendering does not take traditional feature lines as stroke paths. So the discontinuity might be different from previous methods, which often occur at splitting or merging.

3.3 Hair Simulation Model

In this paper, we focus on non-photorealistic representation of the hair animation. Therefore we need a dynamic simulation of hair to generate the hair strands as our input. We use particle-based dynamic system based on Chang et al. [6] to generate the hair strands. The system constructs a single hair strand as several chaining particles, and performs dynamic simulation. Then perform the hair-hair interactions to generate the animated hair stands. In real world, a person can have as many as 100,000 hair bristles, but for artists the whole representation of hairs is more important than a single bristle. Therefore instead of simulating all 100,000 hair bristles, we only use hundreds of hair strands or lesser.

Chapter 4
Hair Stroke Model

In this chapter, we describe our proposed hair stroke model for generating the comic-style hair rendering. This hair stroke model supports enough information for our system to generating the brush stroke effects.

First, the user input initial positions of the particle joints on each hair strand, and the system will automatically construct the hair stroke model for simulating the hair strokes in real-time. The hair animation will be generated according to the dynamic system similar to Chang et al. [6]. Then the animated hair strands will be transformed to the screen-space stroke paths, as shown in Figure 4.1, with 3D-space particle joints information for stroke simulation.

This chapter is organized as follows. In Section 4.1, we describe the model and information in our input animated hair strands. Then we describe the generation of the stroke paths and the brush model we apply in Section 4.2.

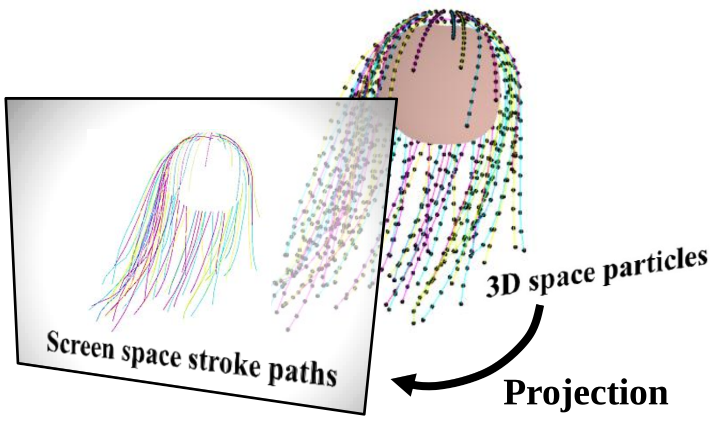
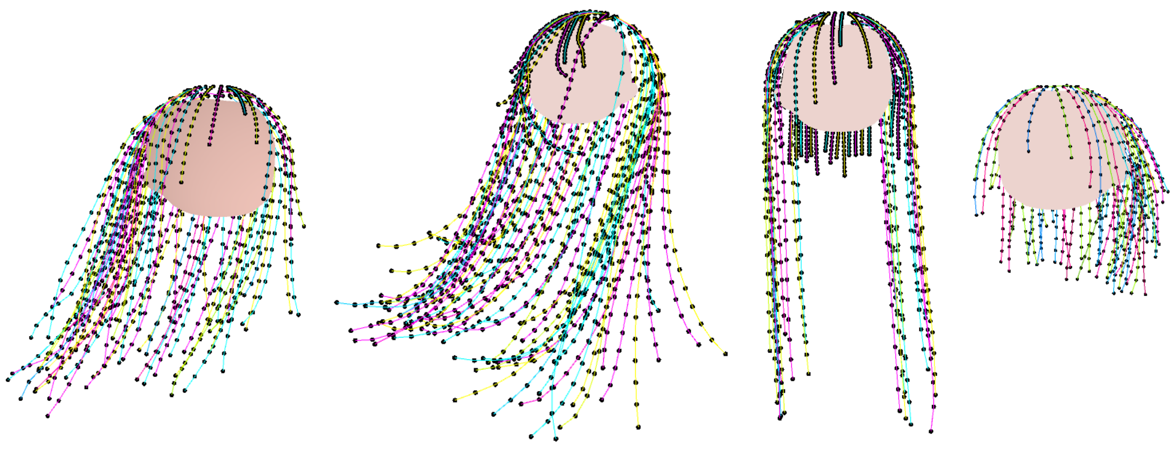


Figure 4.1: Hair strand particles are projected to screen space

4.1 Animated Hair Strands

In this section, we introduce the animated hair strands, which is the input to our system for simulating the hair strokes. The hairstyle is defined at this process by the user to give the initial positions of the particle joints on each hair strand. The hair animation is generated in real-time based on the dynamic particle system proposed by Chang et al. [6]. We offer some simple physical simulations like gravity and wind to generate the animated hair strands. These hair strands will later be used to generate the stroke paths. Besides, each stroke generation will also base on the information provided from the animated hair strand, including the 3D positions, and the neighboring strokes of the same hair strand…etc.

Figure 4.2 shows some animated hair strands models, and the corresponding data information is shown in Table 4.1.



(a) (b) (c) (d)

Figure 4.2: Animated hair strands model

Model	Hair strands	Particles
Figure 4.2 (a)	72	1080
Figure 4.2 (b)	72	1440
Figure 4.2 (c)	84	1600
Figure 4.3 (d)	72	720

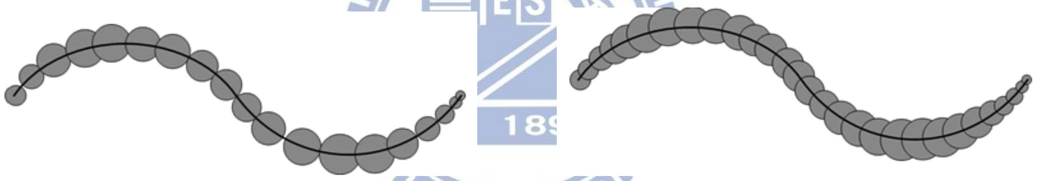
Table 4.1: Data of model in Figure 4.2

4.2 Stroke Brush Model and Path Generation

In this section, we first introduce the stroke brush model for simulating the hair strokes, and describe how to generate the stroke paths from the animated hair strands.

We use the brush models of Weng’s [25] and Ho’s [12] works, which take a single circle to simulating the contact region of brush and paper. Applying these brush strokes on the stroke path will produce the painting stroke that seems like the comic artists drawing the clumpy look of hairs. As shown in Figure 4.3, the dark line is the stroke path, and the circles constitute the brush strokes.

Therefore, the animated hair strands described in the previous section will be transformed to stroke paths. The information provided from each hair strand consists of several 3D space positions of each particle joint on the hair strand. We first project these particles to the screen space, and link them to construct the entire stroke path of each hair strand. For performance consideration, we simply use linear interpolation to generate the stroke path. Although it might sometimes produce sharp angle between hair's segments, if the adjacent particles are too close and the dynamic of hair is violent. It is hard to discover the artifacts.



(a) Stroke on each joint of the hair strand.

(b) Interpolation of each joint to generate the entire hair stroke.

Figure 4.3: The stroke model

Figure 4.3 shows the stroke model and how we generate the stroke paths and apply the brush strokes. Figure 4.3 (a) shows that each brush stroke apply to each particle joint of the hair strand. Later we linearly interpolate the joint strokes to get the entire single hair stroke as show in Figure 4.3 (b).

Chapter 5
Hair Stroke Simulation

In this chapter, we describe the process of brush stroke effect generation. Later we combine all the stroke shaders and render them in a specific order to generate the desired results. Figure 5.1 shows the flowchart of our hair stroke simulation.

In our implementation, the brush stroke effect shaders consist of: base stroke (basic information like size, basic color, and normal of the stroke), silhouette stroke (boundaries and silhouettes effect), cel-shaded stroke (cel-shading effect), highlights stroke (highlights effect), and thin fine hair stroke (some dynamic thin fine hairs). In the rest of this chapter, we introduce a stroke shader in each section. Each shader will generate its own brush stroke effect according to the information of animated hair strands, and apply to the stroke paths.

During rendering, the order of the artistic drawing should be taken into account. We combine all generated strokes and render them in the following order: silhouette strokes, base strokes, diffuse strokes, highlight strokes, and thin fine hair strokes. The depth order should also be ensured. We propose a simple and fast method to combine two orders, as described in in Section 5.2.

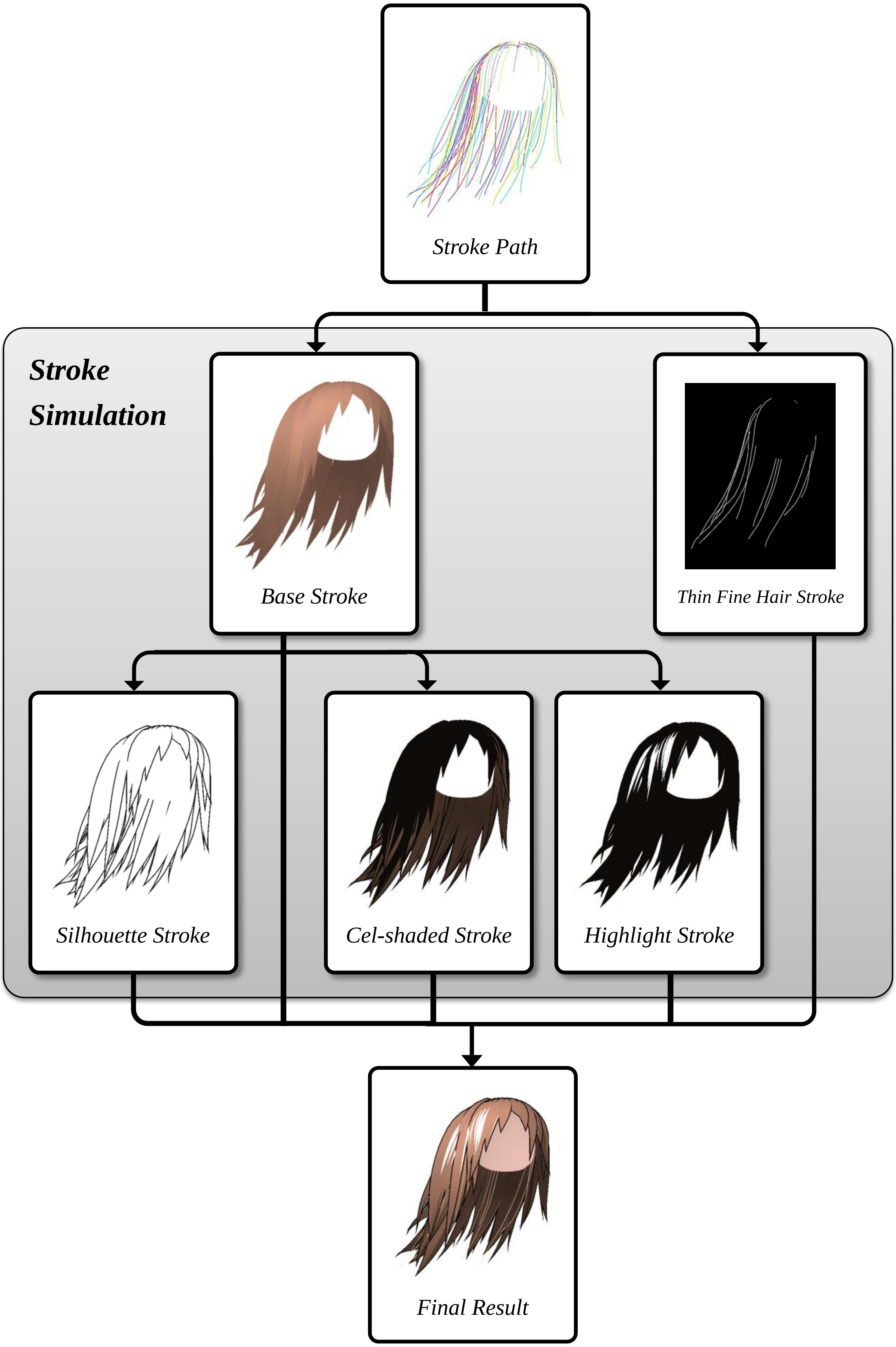


Figure 5.1: The flowchart of stroke simulation

5.1 Base Stroke

In this section, we generate the basic information of a stroke. There are two attributes we need to determine in this step, one is the size of the stroke, the other is the basic color of the stroke. These attributes will also be used in the other shaders.

5.1.1 Stroke Size

To mimic the clumpy look of hairs, first we need to define the size **s** of each hair stroke path, which stands for the thickness or the quantity of the hair clump. Later we need to imitate the shrinking effect when hair stroke path moving toward the hair tail. Here we proposed a method combining two shrinking functions to achieve the desired result.

The first one take a shrinking point x on the hair stroke path as an input, where the size start to shrink. The function is defined as follows:

$$\begin{cases} size_1(i) = \mathbf{s}, & if\ i < x \\ size_1(i) = \dfrac{i-x}{num-x} \cdot \mathbf{s}, & if\ i \geq x \end{cases} \qquad \mathbf{(5.1)}$$

where *i* is a point on the hair stroke path, and *num* stands for the total number of current hair stroke path. This function generates the shrinking effect when a point on stroke path is behind the shrinking point *x*. But, at the shrinking point, it might create a sharp angle. Thus we further introduce another shrinking function which scales the size by the sine function, as shown below:

$$size_2(i) = \sin\left(\left(offset + \frac{i}{num}(1 - offset)\right) \cdot \pi\right) \cdot \mathbf{s} \qquad \mathbf{(5.2)}$$

where *offset* is a displacement of the sine function as shown in Figure 5.2. This function also creates the shrinking effect of the hair strand when moving toward the hair tail. But this effect is not obvious that makes the hair clump seems too thick. This thick effect is not always satisfied, for example the bangs part of hairs. Therefore, we combine the two shrinking function with weighting coefficient to obtain the desired result, as shown below:

$$size_f(i) = weight \cdot size_1(i) + (1 - weight) \cdot size_2(i) \qquad \mathbf{(5.3)}$$

where $size_f$ is the final size of a point on the hair stroke path.

Figure 5.3 shows the effect of the shrinking functions. In order to perturb the size of different hair strokes, the user can apply different weighting values.

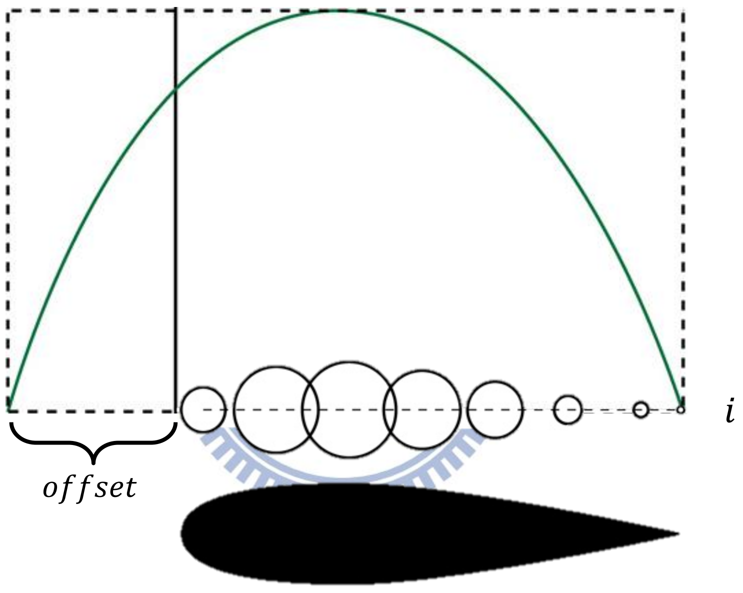
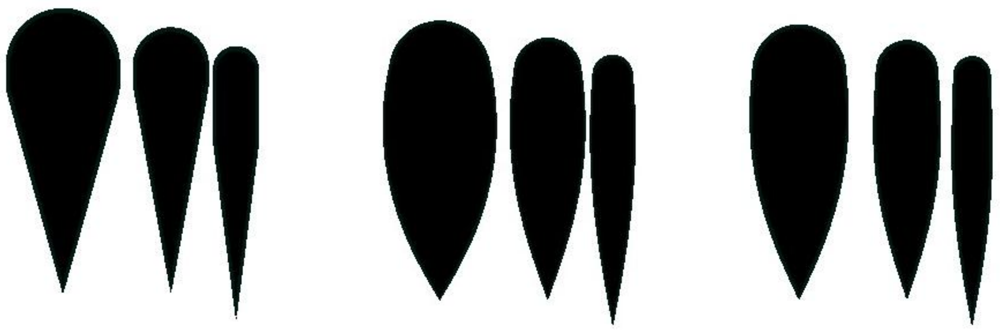


Figure 5.2: Size generated by *size*$_2$



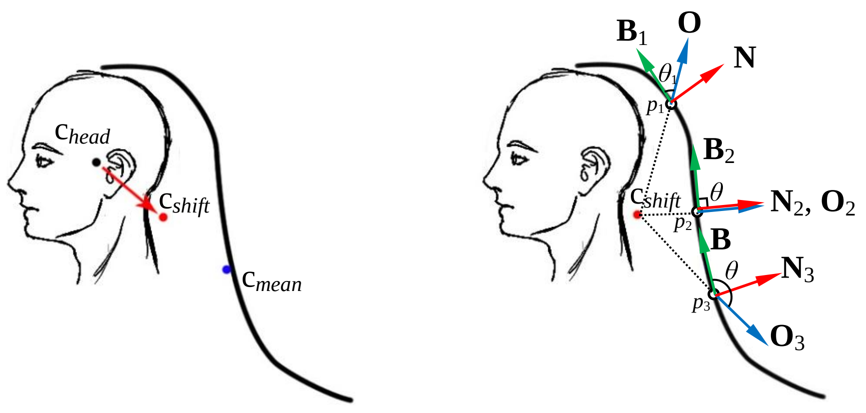
(a) Function *size*$_1$ (b) Function *size*$_2$ (c) Function *size*$_3$

Figure 5.3: Stroke size samples of each function

5.1.2 Basic Color of Stroke

From the comic and cartoon examples in Figures 3.2 and 3.3, we can see that there are two types of color representation of the hair. One is the pure color, and the other is with diffuse effect. Our system provides both types of simulation for all stroke generation. Pure color simulation is simply by setting a single color. Diffuse effect is generated by applying the Phong reflection model $\boldsymbol{l} \cdot \boldsymbol{n}$. The problem is that the hair stroke paths only consist of particles and no any geometry, such as the mesh surface for normal information. Therefore, we propose an alternative way to obtain the approximate normal vector from the hair strand particles.

A simple way is using the head position c_{head} as the center and creating a radial normal vector from head position to each hair strand particle. But there will be a problem when facing to the long hair at the hair tail. Thus, for each hair strand, we first compute the mean position c_{mean} of the whole particles on the hair strand. Then shift the head position c_{head} amount to the mean position c_{mean} to get the new center c_{shift}. The normal vector now is from the shifted center c_{shift} to each hair strand particle. The amount of shifting can be controlled. Figure 5.4 (a) show the shifted center c_{shift}, then we modify the current normal to get more correct result.



(a)Shifted center. (b) Modified normal vector.

Figure 5.4: Normal vector computation

As shown in Figure 5.4 (b), we put three points on the same hair stroke. The outward vector from the shifted center is marked as vector $\mathbf{O}_i$, which is the current normal result. We also give the backward vector $\mathbf{B}_i$ of each point that points backward to the previous point on the same hair stroke. Then we will compute the angle between the outward vector $\mathbf{O}_i$ and the backward vector $\mathbf{B}_i$. If this angle is vertical, the outward vector $\mathbf{O}_i$ is the same as normal vector $\mathbf{N}_i$, for example, at the point p_2 in Figure 5.4 (b). Otherwise, we will modify the outward vector $\mathbf{O}_i$ according to the angle between two vectors to get the normal vector $\mathbf{N}_i$. Consider points p_1 and p_3 in Figure 5.4 (b). The modified function is as follows:

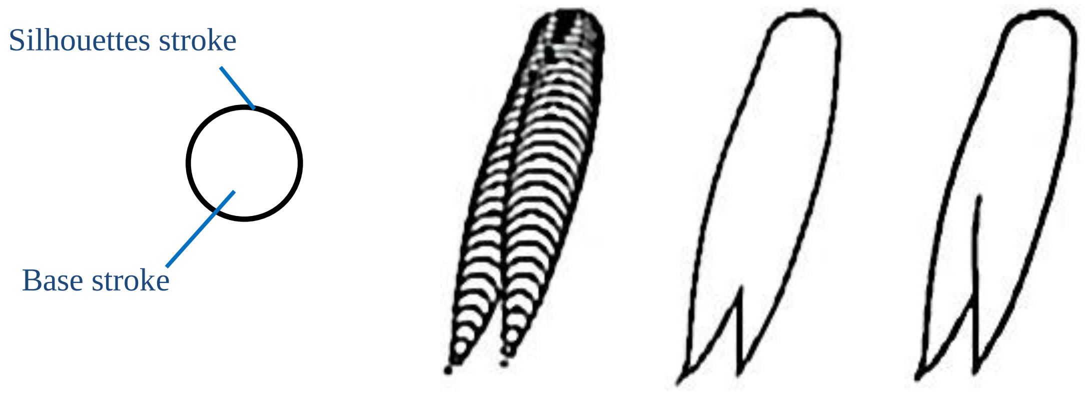
$$\mathbf{N}_i = \mathbf{O}_i - K(\mathbf{O}_i \cdot \mathbf{B}_i)\mathbf{B}_i \qquad (\mathbf{5.4})$$

where the K is the coefficient that controls the amount of modification. Now we can use the Phong reflection model to compute the diffuse color.

5.2 Silhouette Stroke

In this section, we introduce how to generate the silhouette stroke effect from our model without mesh information. Our model takes particles as input to generate the hair stroke paths, which lead to no hair polygon mesh. Therefore traditional methods to find feature lines on the surface cannot be applied to our model. We propose a method to generate the silhouette effect in the following paragraphs..

Using the size information generated in the base shader, we simply enlarge the size to create the boundary lines stroke effect as shown in Figure 5.5 (a). It gets problem when rendering in depth order as shown in Figure 5.5 (b). If we follow the artists' drawing order that first draw silhouette strokes and then base strokes, the final result will only consist of boundary lines as shown in Figure 5.5 (c). Thus we need to combine the depth order and the artists' drawing order to get the desired result as shown in Figure 5.5 (d).



(a) (b) (c) (d)

Figure 5.5: Silhouette stroke model and rendering order problem

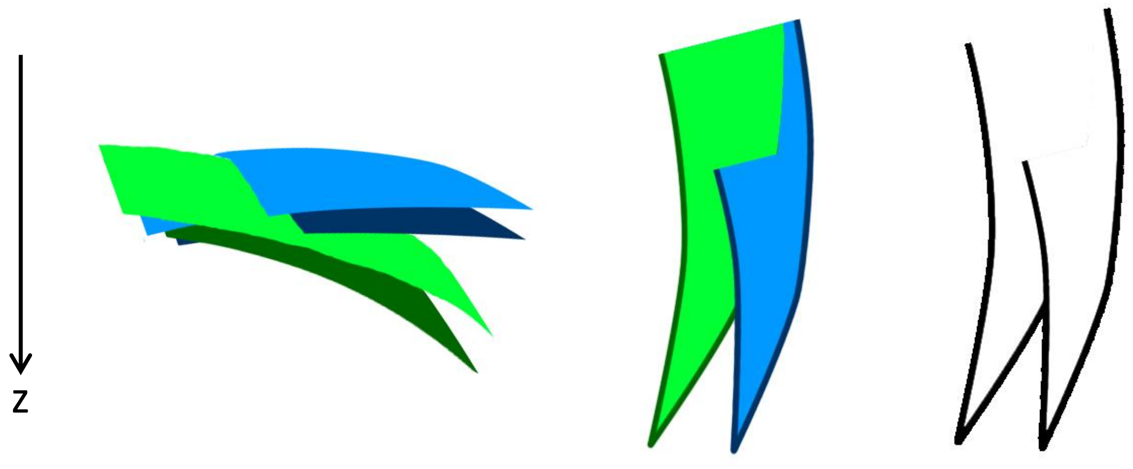
(a) Our silhouette stroke model. (b) Render in depth order.
(c) Render in stroke order. (d) Expected result.

A simple idea to combine the two orders is to render each strand of hair stroke separately. First sort and render the hair stroke strand by strand in depth order, and then render each strand in artists' drawing order. But it may create the popping effect when the strand order changes between contiguous frames. Computing stroke order strand by strand also needs extra time to statistically compute the depth of all particles on the hair strand, which will decrease the performance. Therefore, we use a re-ordering mechanism that adjust each stroke order in depth based on the artists' drawing order, and render them just using depth test, which will be faster and less popping effect.

Each effect shader only generates its own hair stroke effect. From the depth information of hair strands particles, each projected hair stroke can be a piece of surface in space. For a given strand of hair stroke path, it will generate several stroke effects on the same position. Therefore we need to render these stroke effects in the right order to get the desired result.

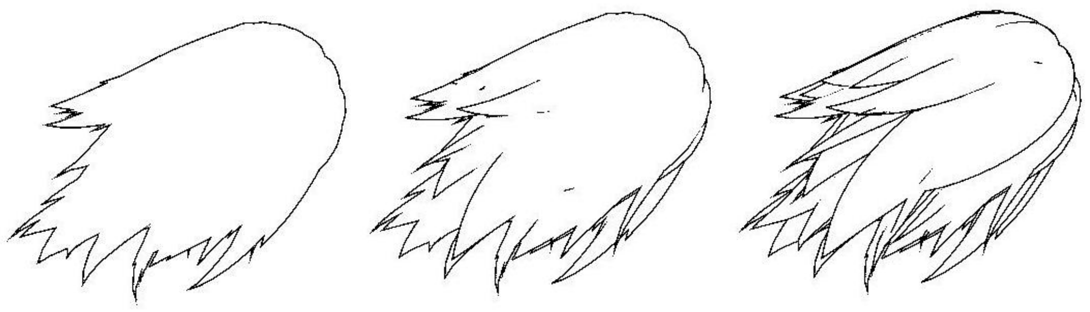
Our method is to give a new depth value to each stroke, and render them using depth test. The base shader gives the basic position, where depth value we do not change. For other shaders, we will change the depth value of the given stroke based on the artists' drawing order with the depth value of the corresponding stroke from base shader. Take silhouette shader as an example, the silhouettes stroke should draw before the other strokes. Thus we apply the new depth value that shift the depth value of the corresponding stroke farther from base shader.

Consider Figure 5.6. The blue and green surfaces are two different strands of hair stroke. The darker ones are the silhouette stroke planes, and the lighter ones are the base stroke planes. The darker one is a little larger than the corresponding lighter one to create the silhouette effect. The darker one is also shifted farther than the lighter one. Figure 5.6 (a) shows the side-view of these four surfaces, and we can see different depth orders at different segments of the stroke path. Figure 5.6 (b) displays the top-view of these four surfaces. Let the lighter ones be the same white color, and the darker ones be the same black color. We can obtain the result as shown in Figure 5.6 (c), which is the silhouette effect we need. The amount of shift can be controlled by the user, which will affect the density of the silhouettes as shown in Figure 5.7.



(a) Depth order of each stroke from side-view. (b) Top-view. (c) Silhouette stroke effect.

Figure 5.6: Re-order strokes example: silhouette strokes simulation



(a) Only boundaries (b) Few silhouettes (c) Dense silhouettes

Figure 5.7: Silhouette stroke density controlled by the depth shifting

5.3 Cel-shaded Stroke

The cel-shaded stroke is similar to the base stroke that also offers both pure color and diffuse color. In this shader, we generate the feathering cel-shaded stroke. The feathering effect is common in comics and cartoon animations, which is often used by the artists to express rich dynamics of hair strands in a clump of hairs. Figure 5.8 gives a simple overview of cel-shaded stroke generation. We first determine the cel-shading region by the gray level information from the shaded hair. Then we generate the feathering effect in the cel-shading region. Finally, we apply color to the strokes based on the basic color generated in the base shader.

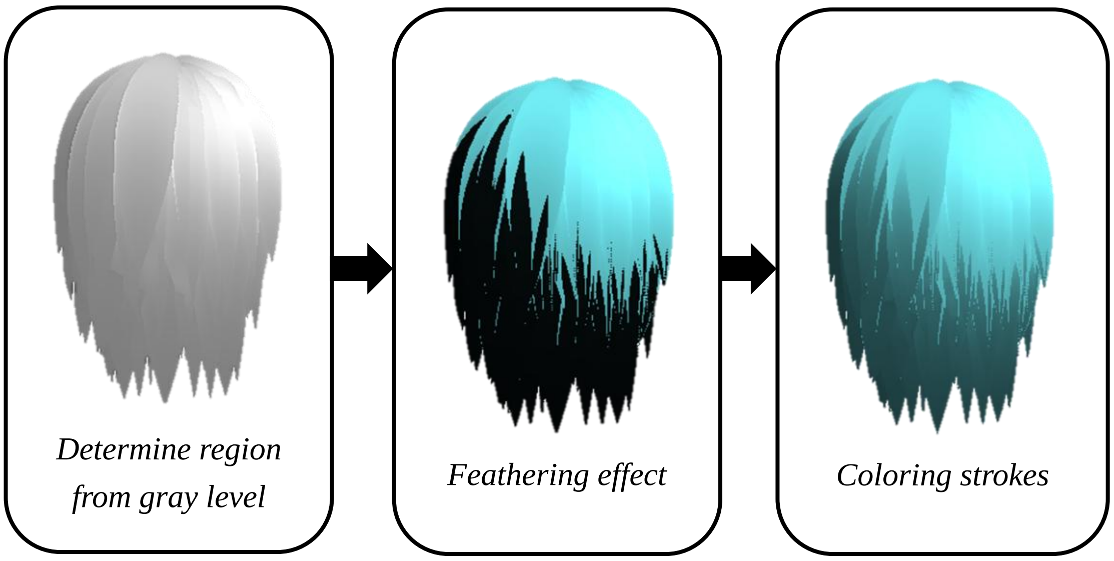


Figure 5.8: The flowchart of cel-shaded stroke simulation

5.3.1 Cel-shading Region and Color

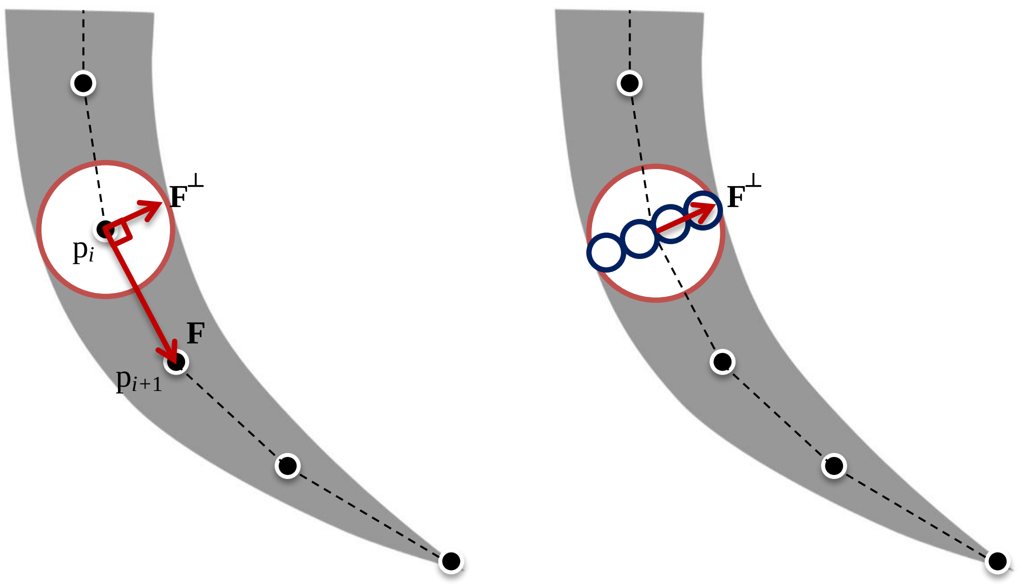
Our method to generate cel-shading effect is similar to traditional methods that adding a threshold to the gray level of shaded model. We use the normal model introduced in the base stroke section to create the gray level of the diffuse shaded stroke based on Phong reflection model. Then use a threshold to determine the cel-shading region. The user can control the threshold to determine the cel-shading region. We also provide perturbation to create more dynamic results. The effects of perturbation are shown in Figure 5.11.

In the cel-shading region, our system offers both pure and diffuse color effects. Both two types of color are defined based on the basic color generated in the base stroke with lighter or darker result, depending on the cel-shading region.

5.3.2 Feathering Effect

Before generating the feathering effect, we first determine where to apply the strokes within the cel-shading region of the stroke path. The feathering effect is spread on the clump of hairs. Thus these strokes are not just applied on the stroke paths that generated from the animated hair strands.

As shown in Figure 5.9 (a), given a hair strand stroke with basic size information, we first project the particles of the hair strand to the screen space. For a given particle p_i in the cel-shading region, we compute the forward vector **F** pointing to the next particle p_{i+1} on the same hair strand. Then we will generate the stroke position at the direction vertical to the vector **F,** as the blue circles shown in Figure 5.9 (b). The number of the feathering strokes could also be controlled by user.



(a) Compute the positions direction (b) Create positions at the direction

Figure 5.9: Position of cel-shaded stroke path

We have determined the stroke color and position. Now we need to compute the size of all strokes in the cel-shading region to create the feathering effect. Our feathering stroke size depends on the diffuse gray level and the basic size of the base stroke. First we determine the basic size of the each feathering stroke. For a given hair strand stroke, the number of the feathering strokes in the cel-shading region is define as $N_{feather}$. The basic size of each feathering stroke is simply computed by the following equation:

$$\text{Size}_{feather}^{\text{basic}}(i) = \frac{\text{Size}_{base}(i)}{\text{N}_{feather}} \qquad (\mathbf{5.5})$$

where Size_{base} is the basic size from the base stroke.

For creating feathering effect, we further introduce another gray level threshold ε_{full} which is dependent on the threshold ε_{cel} in determining the cel-shading region. For example, if the cel-shading region generates the darker color than the base stroke, then this threshold should be darker. If the cel-shading region has lighter color, then it should be much lighter. Within this region, the sizes of the feathering strokes are the basic size defined by the Equation 5.5. Outside the region, the sizes shrink to zero when reaching the cel-shading boundary. The shrinking size is computed based on the gray level of the current stroke as follows:

$$\begin{cases} \mathrm{Size}(i) = \mathrm{Size}^{\mathrm{basic}}_{feather}(i), & if\ gray(i) \le \varepsilon_{full} \\ \mathrm{Size}(i) = \dfrac{1 - |gray(i) - \varepsilon_{full}|}{|\varepsilon_{cel} - \varepsilon_{full}|} \cdot \mathrm{Size}^{\mathrm{basic}}_{feather}(i), & if\ \varepsilon_{full} < gray(i) \le \varepsilon_{cel} \end{cases} \quad (\mathbf{5.6})$$

where $gray(i)$ is the gray level of the stroke i. The equation $gray(i) \le \varepsilon_{full}$ means that stroke i is within the threshold region or not. Figure 5.10 shows the effect of shrinking. Gray stroke gives the gray level information and the size of base stroke. We can easily find that these two thresholds separate the regions with different sizes of the cel-shaded stroke. The system also provides perturbation for both thresholds. In Figure 5.11, we can see the difference whether the perturbation is applied or not.

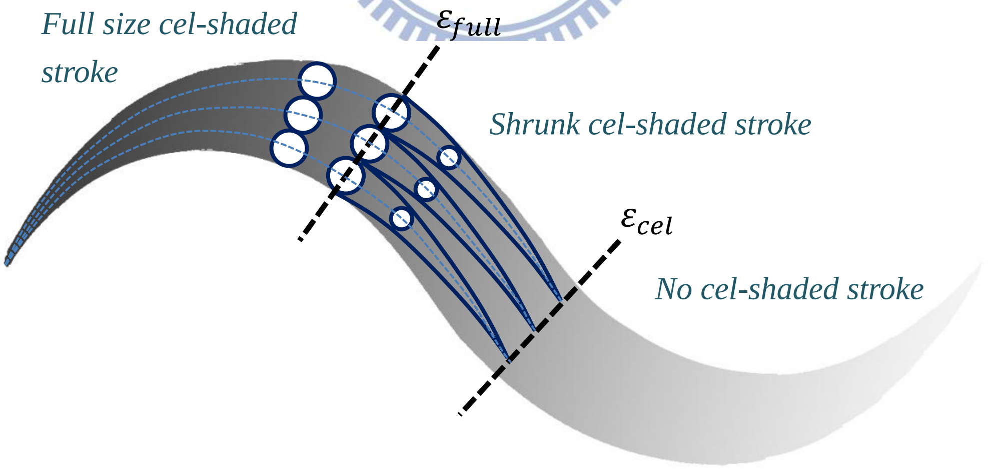
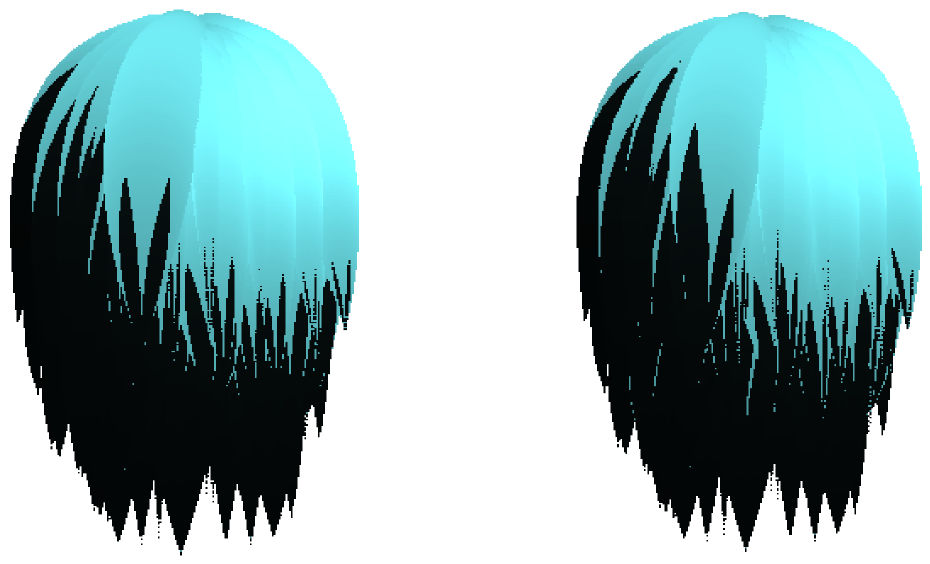


Figure 5.10: Size of cel-shaded stroke



(a) Cel-shading region (no perturbation)   (b) Cel-shading region (with perturbation)

Figure 5.11: Perturbation for feathering effect

5.4 Highlight Stroke

Highlight effects can be classified in two types: one is like Figure 3.1 that the highlight stroke is long or even spread to the hair end; the other is like Figure 3.2 (a) and Figure 3.3 (b), which is only at small region and feel like specular effect. Our system provides both effects for the users.

The second effect is simple and generated in the way like the cel-shaded stroke, while we replace the gray level of diffuse shaded model with the specular term of the Phong reflection model. Two thresholds are also needed to determine the region and create the feathering effect.

Using gray level of the specular to generate the highlight effect only consists of small region. Thus, to achieve the highlight effect that spread on the hair stroke path, we need another method to decide the region. We compute the distance between light and the stroke position to decide the stroke size. Figure 5.12 shows the relation of stroke size and the distance to the light. Two thresholds are also needed here to generate the feathering effect. Figure 5.13 shows the two different highlight effects. The color of the highlight stroke is white by default.

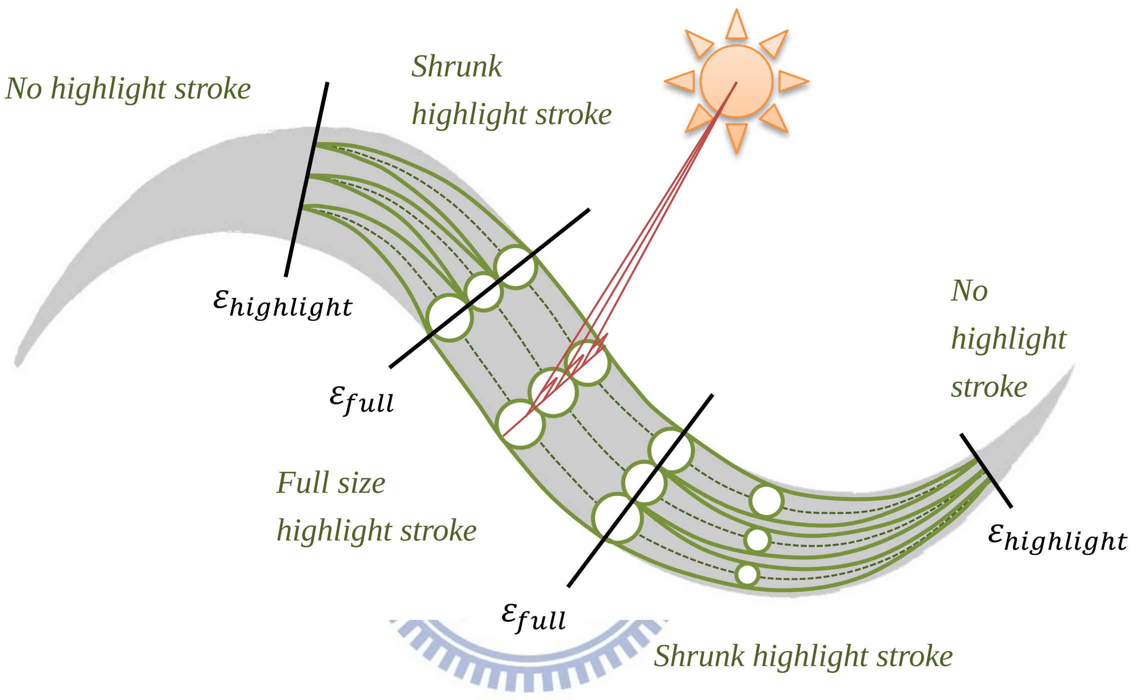
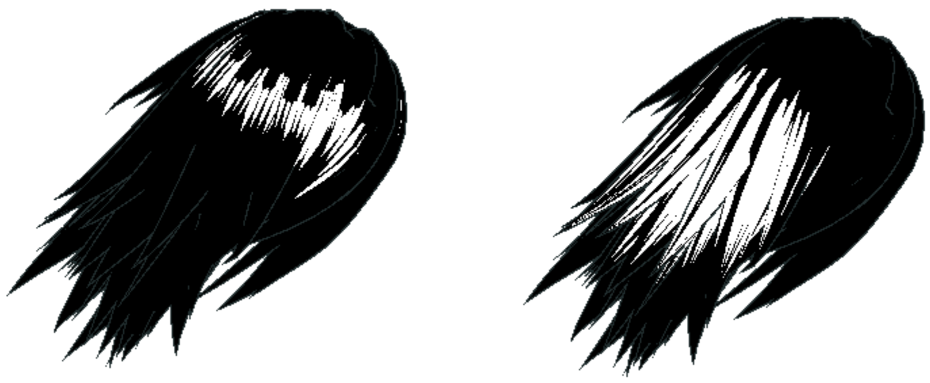


Figure 5.12: Size of highlight stroke



(a) Highlights in specular region. (b) Highlights spread through whole hairs.

Figure 5.13: Two types of highlight stroke

5.5 Thin Fine Hair Stroke

The fine hair stroke is a special stroke and its properties are different from the other stroke effects. Although artists often draw clumpy look of hairs, some of them will also add thin lines to represent fine hairs expression. These effects are usually added at the final step of painting that gives the hair result more dynamic and attractive.

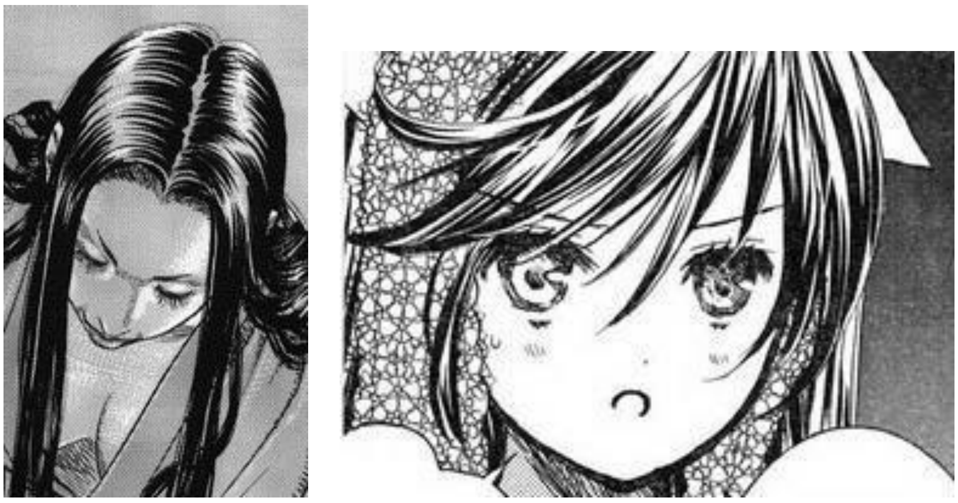
So far, the stroke shaders we introduce are all bounded in the clumpy region, which is the basic size generated from the base strokes. Besides, the stroke paths are the same or derived from the base strokes. The thin fine hair strokes are different that they use their own stroke path and might be drawn across several clumps of hairs. The stroke paths for the thin fine fair strokes are also generated from the animated hair strands. But the stroke path will be only applied to the thin fine hair strokes. We generate the thin fine hair stroke just rendering the stroke path as a thin line, and of course the user can control the thickness of the hair strokes. The color of the thin fine hair strokes is usually lighter than the other strokes. Thus our system sets the color lighter than the basic color from the base stroke. But the user still can control and change the color as they desired.

Chapter 6
Results and Discussion

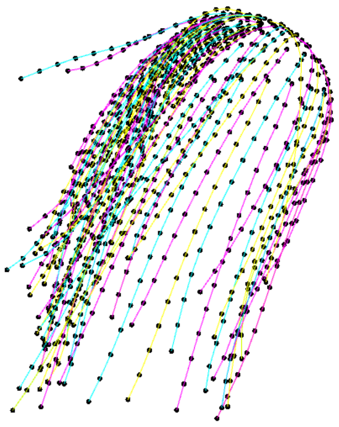
In this chapter, we present our implementation results. The input source is 3D particles of animated hair strands, and the output results are rendered with different combination of stroke shaders. The algorithm is implemented in C++ Language using OpenGL. The stroke simulation process is implemented in GPU by using GLSL. The experiment is carried out on an Intel® Core™ i7 PC with 3GHz CPU and 12GB memory, and the GPU is NVIDIA GeForce GTX 580.

For each example results, we first introduce the reference artist-drawn character. Then we follow and implement the stroke effects of the reference artwork by using our system with several novel views and lighting directions. The dynamics of hairs is either generated by our physical simulation, or adjust manually. And the facial features in the result images are synthesized by post process. Our system can generate stroke effects of dynamic hairs in real-time. Table 6.1 gives the basic data structures of the example models, and the performance of each example.

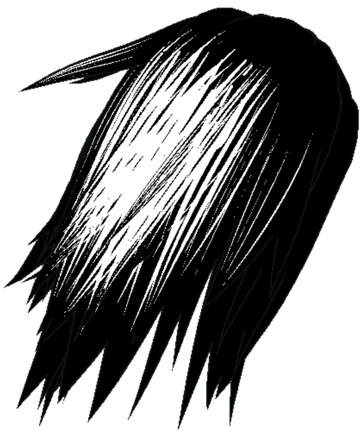
In the first example, we mimic the black and white comics as shown in Figure 6.1 (a). Figure 6.1 (b) shows our proposed hair stands model. The reference artwork consists of highlight stroke only, thus we set base stroke color to pure black and activate the highlight stroke shader to get the result as shown in Figure 6.1 (c). Figure 6.1(d) gives more results of the same effect but with novel view, different lighting, and different parameters.



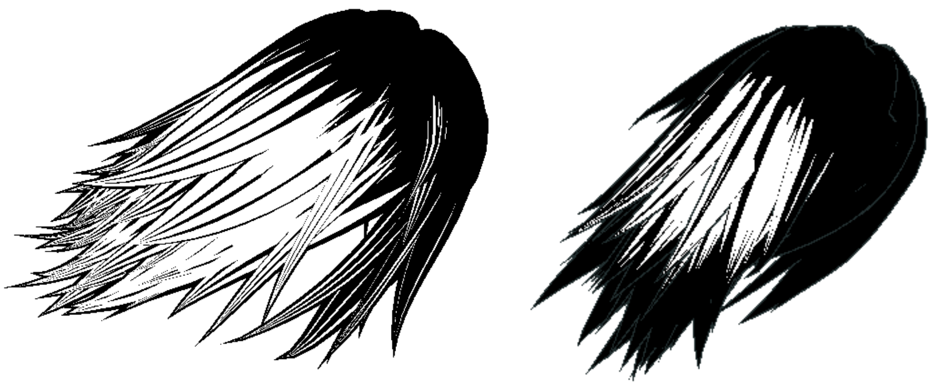
(a) Reference artworks.
Left: artwork by Takehiko Inoue
Right: artwork by PEACH-PIT



(b) Hair strands model



(c) Final result

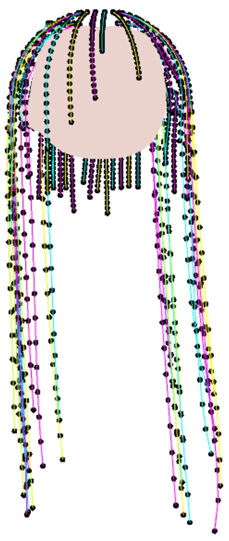


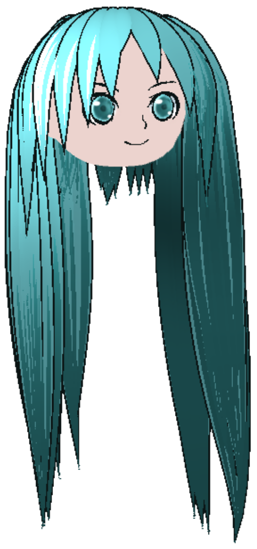
(d) More results with different views and lightings

Figure 6.1: Result with base and highlight strokes

In the second example, we create a hair strands model with two ponytails as shown in Figure 6.2 (b), which is like the famous virtual singer in Japan “Hatsune Miku.” Figure 6.2 (a) shows the reference artwork. In this example, we use the base, silhouette, cel-shaded, and highlight strokes to construct the final image as shown in Figure 6.2 (c) and (d). The coloring result contains diffuse effect which is different from the first example in Figure 6.1 that only contains the pure color

(a) Reference artwork
from Cryton Future Media


(b) Hair strand model


(c) Final result

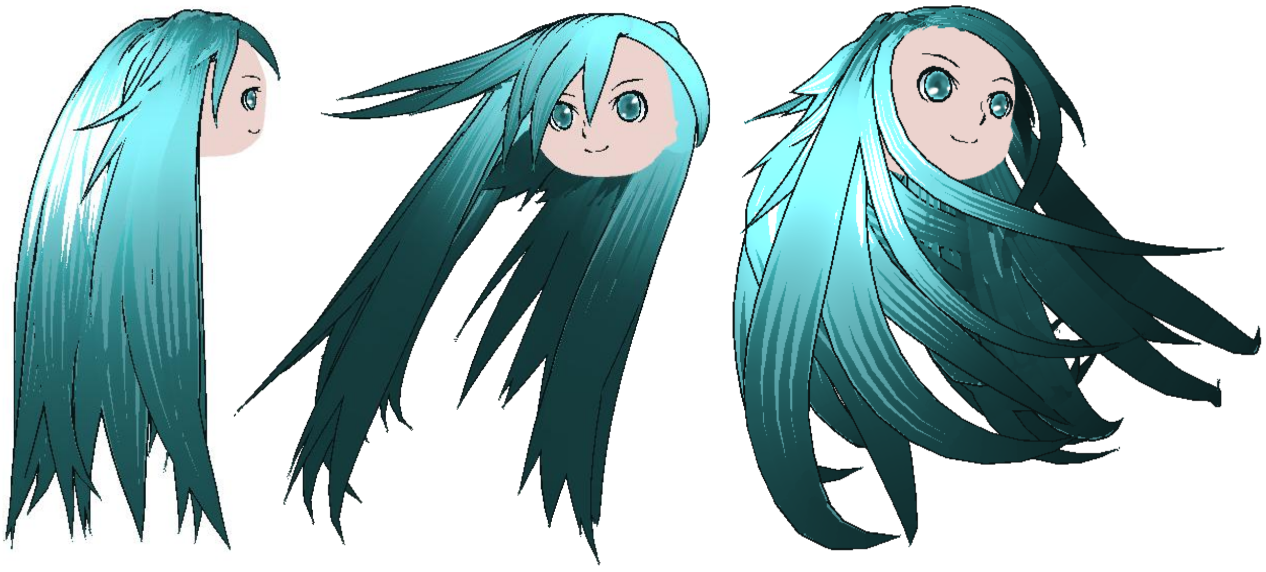
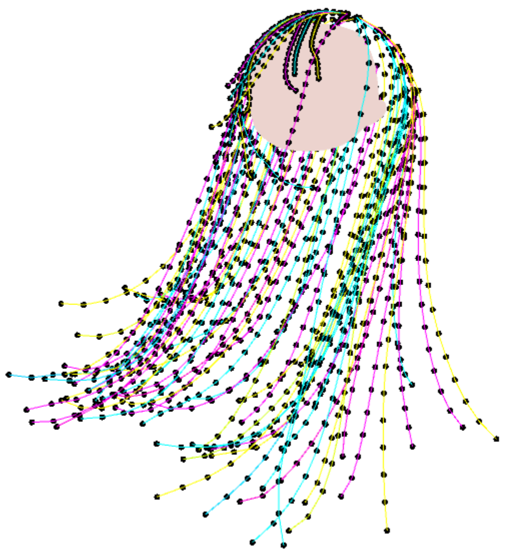

(d) More results with different views and lightings

Figure 6.2: Result with base, silhouette, cel-shaded, and highlight strokes

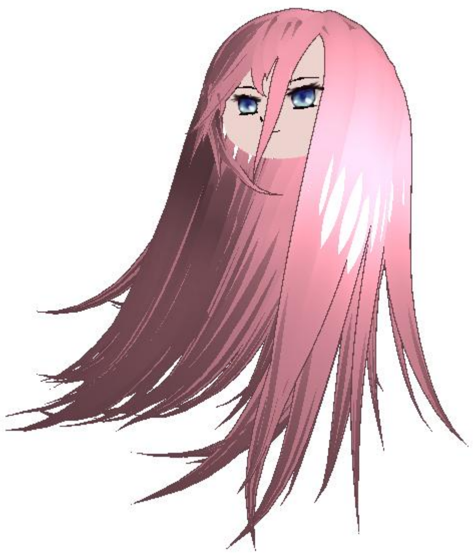
Figure 6.3 is the simulation of long hair. Our reference artwork is a virtual singer "Megurine Luka," as shown in Figure 6.3 (a). Figure 6.3 (b) is the hair strands model. Figure 6.3 (c) and (d) show the result with base, silhouette, cel-shaded, and highlight strokes. We use the same hair strand model to mimic the artwork shown in Figure 6.4 (a), which contains thin fine hair and with pure color. Results are shown in Figure 6.4 (b) with base, cel-shaded, and thin fine hair strokes. We can find that by applying different stroke effect shaders, even with the same hair strands model, our system can still generates different attractive visual effects.

紅一葉
akahitoha

(a) Reference artwork from Cryton Future Media



(b) Hair strands model



(c) Final result

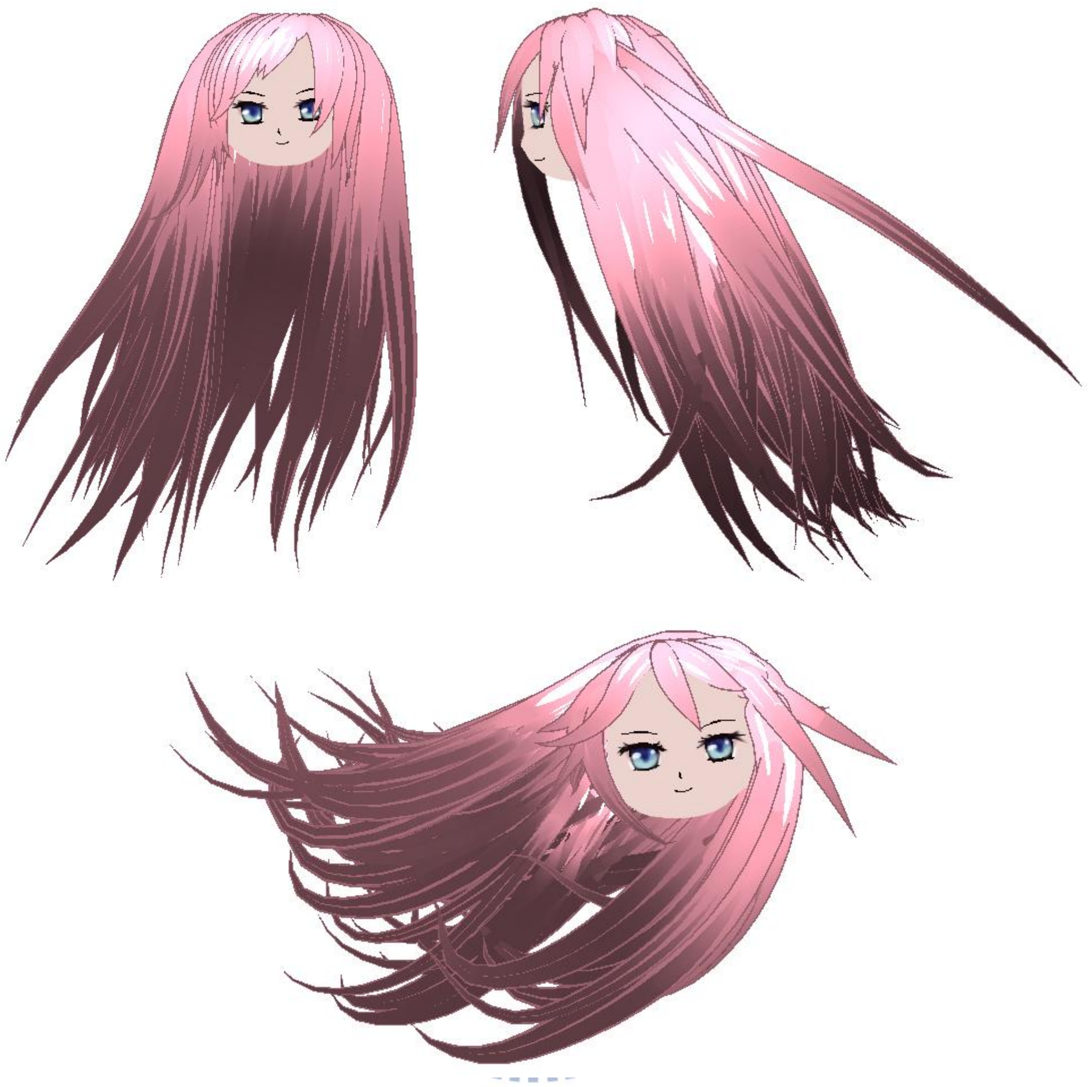
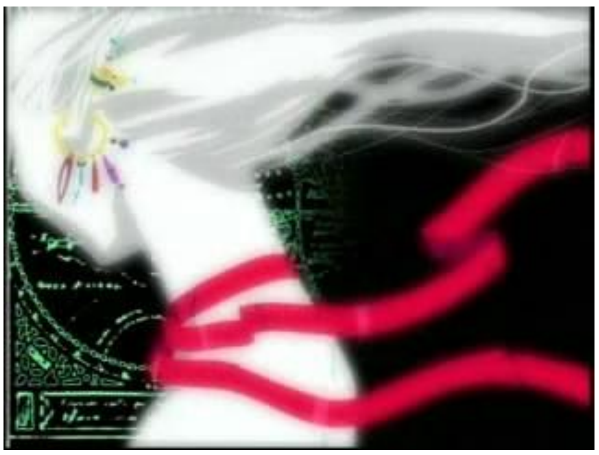
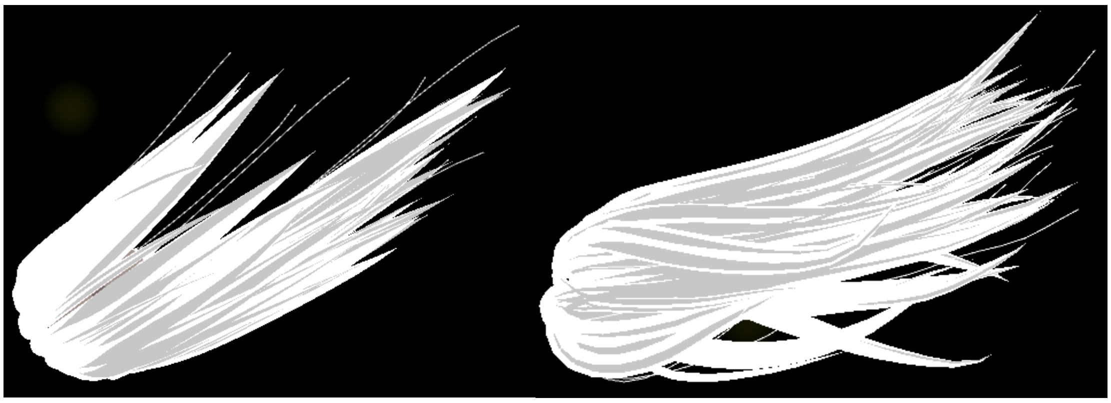

(d) More results with different views and lightings

Figure 6.3: Result with base, silhouette, cel-shaded, and highlight strokes

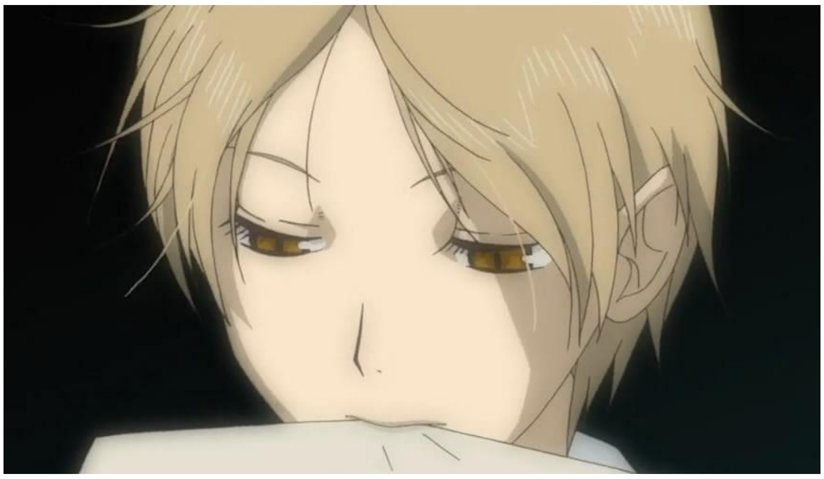

(a) Reference artwork from Animation “Record of Lodoss War OVA (1990)”



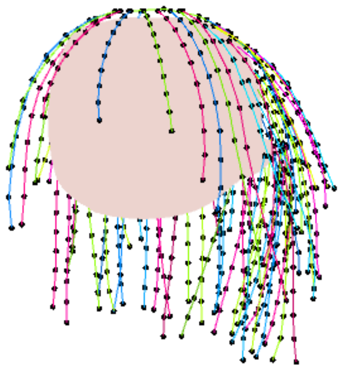
(b) Final results

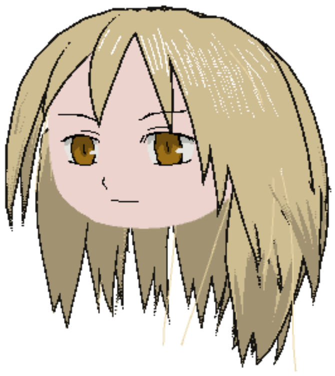
Figure 6.4: Result with base, cel-shaded, and thin fine hair strokes

The last example is the simulation of short hair as shown in Figure 6.5. The reference artwork contain base and cel-shaded strokes with pure color, highlight strokes, and thin fine hair strokes. The highlight strokes in this example are special that they are represented as thin line, thus we further modify the shape of the highlight strokes. Our results are shown in Figures 6.5 (c) and (d). Our system can easily change style by applying different stroke shape as well.



(a) Reference artwork from Animation “Natsume's Book of Friends”


(b) Hair strands model


(c) Final results

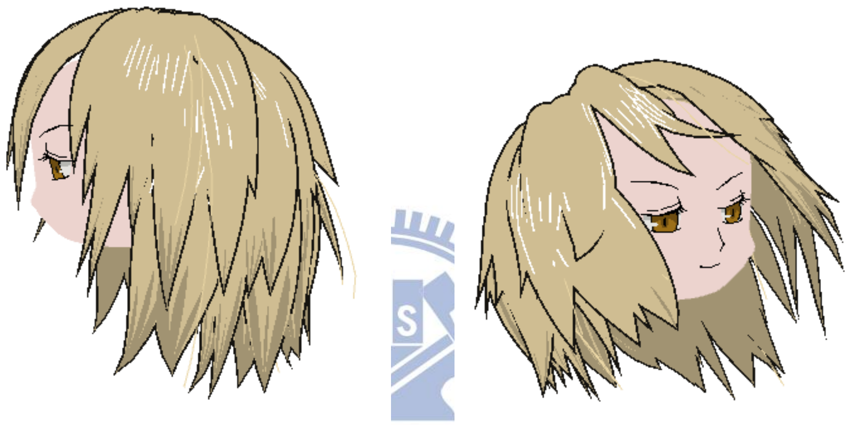

(d) More results with different views and lightings

Figure 6.5: Result with base, silhouette, cel-shaded, highlight, and thin fine hair strokes

Example	Hair strands	Particles	Strokes	Resolution	Performance
Figure 6.1	72	1080	≈ 64k	960x720	30~50 fps
Figure 6.2	84	1600	≈ 160k	960x720	20~30 fps
Figure 6.3	72	1440	≈ 120k	960x720	20~40 fps
Figure 6.4	72	1440	≈ 43k	960x720	30~50 fps
Figure 6.5	72	720	≈ 19k	960x720	30~60 fps

Table 6.1: Statistics of results

Our system runs in real-time and the performance is shown in Table 6.1. The performance is highly related to the number of the generated strokes. As shown in Table 6.1, we only need few hair strands and particles to generate the strokes. Thus the physical simulation can be very fast. The number of generated strokes depends on the interpolation number. For smooth result, the thinner hair strokes should get more interpolation number, like cel-shaded and highlight strokes. Because we generate strokes in screen-space, the larger strokes will contain more pixels and consume more time. But the interpolation number is fewer. More activated stroke effect shaders will also generate more strokes and need more time. That is why examples of Figure 6.2 and 6.3 is much more time consuming than the examples of Figures 6.1 and 6.4. Our system is not fully optimized currently. That is, the performance can be much better.

In most of cases, our results perform well both on quality of stroke simulation and temporal coherence. Actually we get popping effect when the depth order of hair strands changes rapidly due to the re-ordering mechanism. We can apply more steady method to deal with the depth order and the stroke order. But it would consume more time. We believe our re-ordering mechanism achieve a good trade-off between the performance and the temporal coherence. It is fast, and most of time generates the high quality results. However, there is still a limitation of current re-ordering mechanism. One can see that our strokes get mess at the top of head from the previous examples. Our re-ordering mechanism is based on changing the depth value, and the changed value is not constant for the whole hair strands. There are two cases we should preserve first, one is the strokes are on the same hair strand, and the other is the strokes are very close. For these two cases, we believe that the corresponding strokes should be continuously linked together, or it will generate the broken strokes result like mentioned before in Figure 5.5 (b). Therefore the re-ordering values applied on these strokes will be larger than others to produce the continuous entire strokes. Unfortunately these two cases happen too frequently at the top of the head, which will produce too many strokes at there and get mess in that small region.

Compare to previous methods, we believe our result perform well on both stroke quality and performance. Cote et al. [7] took images as input and generated high quality comic-style results. But their method to generate the animation needs individually construct frames by frames, which is time consuming. Noble et al. [18] used hair strands to construct NURBS volume to mimic the clumpy look of hair. Shin et al. [23] also took hair strands as input, and apply billboard texture to generate the carton-style results. We use hair strands as well, but project them to screen space as stroke paths to generate more stroke effects than previous methods. From the result examples above, our system is more flexible than previous methods. By applying different parameters to each stroke shader, we can create several different comic-style results which are similar to the artist drawings.

Chapter 7
Conclusion and Future Work

In this thesis, we propose a system that generate the high quality comic and cartoon style hairs with temporal coherence in real-time. To preserve quality, we propose a novel idea that combines the animated hair strands and the stylized line-based method, which is one of the famous stroke-based methods in NPR, to imitate the artistic strokes. To ensure performance, we analyze and parallelly separate the artistic stroke effects into different shaders performing on GPU in real-time. When rendering, we use a re-ordering mechanism that combines the depth order and the artist-drawn stroke order, which is fast and preserve temporal coherence. Our system is flexible that user can choose which shader effects to be activated and control the parameters of each activated shader to create rich attractive results, even with the same input hair strands model.

Our system now only offers straight hair with different length. It would be interesting if the system could generate more hair style and then apply our stroke simulation. By the combination of animated hair strands and stroke-base method, we could change the hair style in two ways. We could change the physical model of particles in animated hair strands, adding torsion or elasticity to the hair strands to create other types of hair, to generate different stroke paths. If we have torsion information, we may obtain the correct normal instead of the current approximate normal model. The other way is to change the stroke shape, which is one of the benefits of stroke based methods. Given the stroke path, we can apply different shapes of the stroke to create different hair strokes. Figures 7.1 and 7.2 shows the reference artworks and our results by applying different shape of brush stroke. With semicircle brush stroke applied, our system can create flat effect at the hair tails.

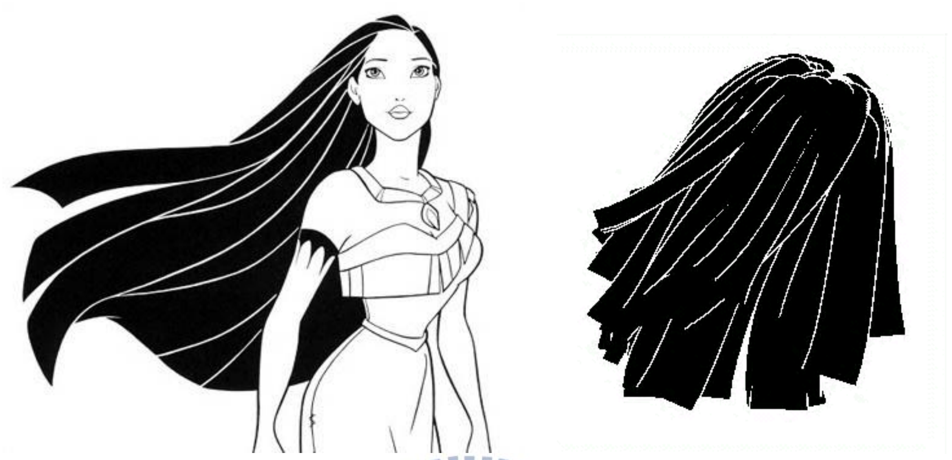

(a) Reference artwork of Disney "Pocahontas" (b) Our result with semicircle brush stroke

Figure 7.1: Result with different brush shape

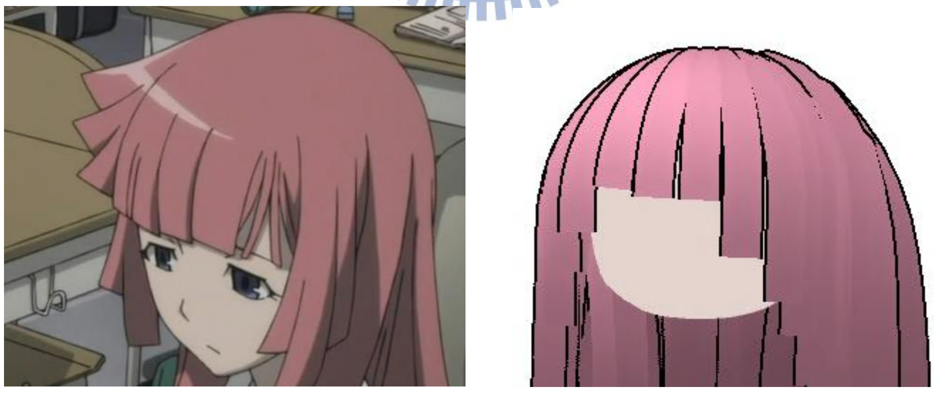

(a) Reference artwork from animation "Star Driver: Kagayaki no Takuto" (b) Our result with semicircle brush stroke

Figure 7.2: Result with different brush shape

An interesting idea is combine the 3D painting techniques like Schmid et al. [21] or Katanics et al. [16]. The art designers draw hair strokes directly on the screen just like they do digital painting. The system will sample the strokes as particles of hair strands and apply dynamic simulation to generate novel stroke paths. The artist-drawn strokes will be also analyzed automatically for stroke simulation including color, size, cel-shading region and color, highlight… etc. for each shader. The design and creation of stylized hair will become much easier and more direct.

References

[1] BARAN, I., SCHMID, J., SIRGRIST, T., GROSS, M., AND SUMNER, R. 2011. Mixed-order compositing for 3D paintings. In Proceedings of ACM SIGGRAPH Asia.

[2] BÉNARD, P., BOUSSEAU, A., AND THOLLOT, J. 2011. State-of-the-art report on temporal coherence for stylized animations. Computer Graphics Forum 30, 8, 2367-2386

[3] BÉNARD, P., COLE, F., GOLOVINSKIY, A., AND FINKELSTEIN, A. 2010. Self-similar texture for coherent line stylization. In Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering (NPAR).

[4] BÉNAED, P., LAGAE, A., VANGORP, P., LEFEBVRE, S., DRETTAKIS, G., AND THOLLOT, J. 2010. A dynamic noise primitive for coherent stylization. Computer Graphics Forum 29, 4, 1497-1506

[5] BUCHHOLZ, B., FARAJ, N., PARIS, S., EISEMANN, E., AND BOUBEKEUR, T. 2011. Spatio-temporal analysis for parameterizing animated lines. In Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering (NPAR).

[6] CHANG, J., T., JIN, J., AND YU, Y. 2002. A practical model for hair mutual interactions. In Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation.

[7] COTE, M., JODOIN, P.-M., DONOHUE, C., AND OSTROMOUKHOV. 2004. Non-photorealistic rendering of hair for animated cartoons. In Proceedings of GRAPHICON' 04.

[8] DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. ACM Trans. Graphics 22, 3.

[9] DECARLO, D. AND RUSINKIEWICZ, S. 2007. Highlight lines for conveying shape. In

Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering (NPAR).

[10] GRABLI, S., TURQUIN, E., DURAND, F., AND SILLION, F. 2010. Programmable rendering of line drawings from 3D scenes. ACM Trans. Graphics 29, 2, 18.

[11] HERTZMANN, A. 2003. A sirvey of stroke-based rendering. Compter Graphics and Applications, IEEE 23, 4, 70-81.

[12] HO, Y.-W., AND SHIH. Z.-C. 2005. The synthesis of Chinese fine-brushwork painting for flower. Master thesis, National Chiao Tung University.

[13] JUDD, T., DURAND, F., AND ADELSON, E. 2007. Apparent ridges for line drawing. ACM Trans. Graphics 26, 3, 19.

[14] KALNINS, R. D., DAVIDSON, P. L., MARKOSIAN, L., AND FINKELSTEIN, A. 2003. Coherent stylized silhouettes. ACM Trans. Graphics 22, 3.

[15] KASS, M., AND PESARE, D. Coherent noise for non-photorealistic rendering. 2011. In Proceedings of ACM SIGGRAPH.

[16] KATANICS, G., AND LAPPAS, T. 2003. Deep Canvas: Integrating 3D Painting and Painterly Rendering. In Theory and Practice of Non-Photorealistic Graphics: Algorithm, Methods, and Production System, ACM SIGGRAPH 2003 Course Notes.

[17] KOLOMENKIN, M., SHIMSHONI, I., AND TAL, A. 2008. Demarcating curves for shape illustration. ACM Trans. Graphics 27, 5, 1–9.

[18] NOBLE, P., AND TANG, W. 2004. Modeling and animating cartoon hair with NURBS surfaces. In Computer Graphics International (CGI), 60-67

[19] OHTAKE, Y., BELYAEV, A., AND SEIDEL, H. 2004. Ridge-valley lines on meshes via implicit surface fitting. In Proceedings of ACM SIGGRAPH, 848-855.

[20] ROMAIN, V., DAVID, V., JIAZHOU, C., PASCAL, B., XAVIER, G., AND CHRISTOPHE, S. 2011. Implicit Brushes for Stylized Line-based Rendering. In Proceedings of EUROGEAPHICS.

[21] RUSINKIEWICZ, S., COLE, F., DECARLO, D., AND FINKELSTEIN, A. 2008. Line drawings from 3D models. In Proceedings of ACM SIGGRAPH Course Notes.

[22] SCHMID, J., SENN, M., GROSS, M., AND SUMNER, R. 2011. OverCoat: an implicit canvas for 3D painting. In Proceedings of ACM SIGGRAPH.

[23] SHIN, J., HALLER, M., MUKUNDAN, R. 2006. A stylized cartoon renderer. In ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, 14-16

[24] WARD, K., BERTAIL, F., KIM, T. Y., MARSCHNER, S. R., CANI, M., P., AND LIN, M., C. 2007. A survey on hair modeling: styling, simulation, and rendering. IEEE Trans. Visualization and Computer Graphics 13, 2, 213-234

[25] WENG, S.-Z., SHIH, Z.-C., AND CHIU, H.-Y., 1999. The synthesis of Chinese ink painting. National Computing Symposium'99, 461-468.

[26] XIE, X., HE, Y., TIAN, F., SEAH, H. S., GU, X., AND QIN, H. 2007. An effective illustrative visualization framework based on photic extremum lines (PELs), IEEE Trans. Visualization and Computer Graphics 13, 6, 1328-1335

[27] YUKSEL, C., SCHAEFER, S., AND KEYSER, J. 2009. Hair meshes. In Proceedings of ACM SIGGRAPH Asia.

[28] YUKSEL, C., AND TARIQ, S. 2010. Advanced techniques in real-time hair rendering and simulation. In Proceedings of ACM SIGGRAPH Course Notes.

[29] ZAHNG, L., HE, Y., XIA, J., XIE, X., AND CHEN, W. 2011. Real-time shape illustration using laplacian lines. IEEE Trans. Visualization and Computer Graphics 17, 7, 993-1006