Master Thesis:
Hair Simulation & Rendering

Mike Lansink (0088714)

Human Media Interaction

Department of Computer Science

University Twente

Graduation Committee:

Dr. Job Zwiers

Dr. Mannes Poel

Ir. Herwin van Welbergen

October 2009 – June 2010

7522 NB Enschede

The Netherlands

Master Thesis:
Hair Simulation & Rendering

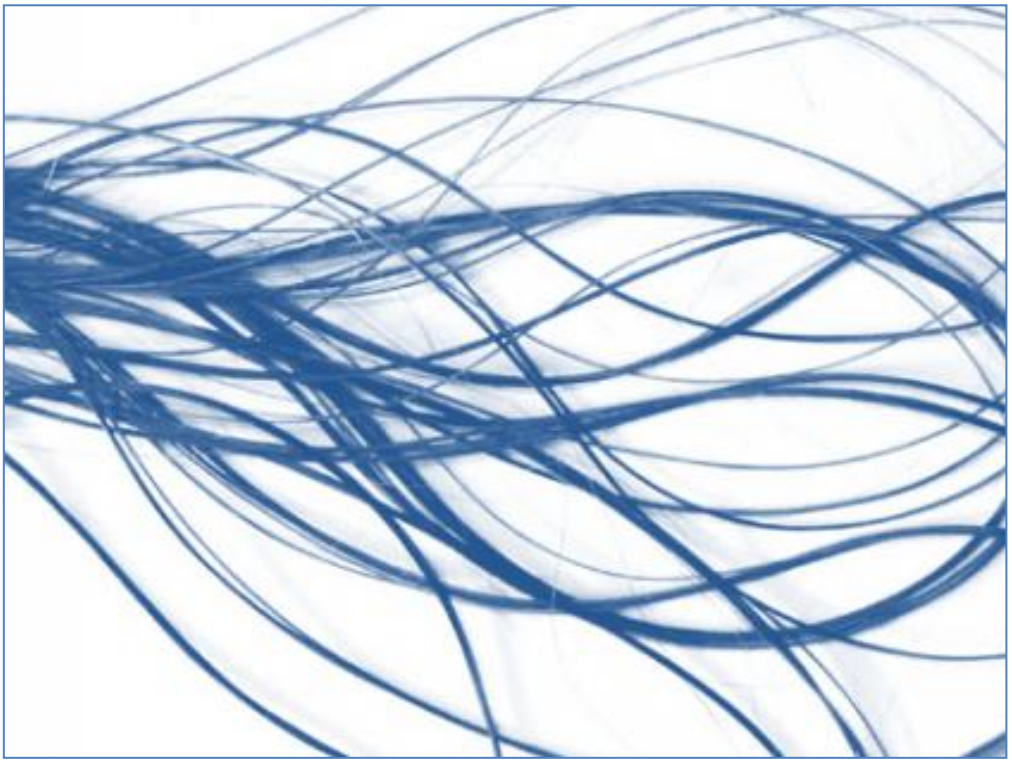

Table of Contents

List of Figures

List of Tables

Abstract

In the fields of Computer Graphics and Human Simulation, hair offers one of the most challenging physics and rendering problems. Correctly modelling the shape and the appearance of hair is crucial if one wants to generate realistic looking characters. Many of the problems are caused due to the intrinsic properties of natural human hair.

In this Master Thesis I have addressed this problem by combining existing models and techniques into a configurable framework which allows the user to generate the intended hair coupe. The Super-Helix model – (Bertails, et al., 2005b) – in combination with Rosenbrock's optimization method – (Rosenbrock, 1960) – have resulted in a physical hair model which is able to correctly simulate the overall shape of a hair coupe at rest. This physic model takes into account, both, the non-linear elastic properties of a individual hair strand and the gravitation field pulling at the hair strand. The visual aspect of the hair coupe are accounted for by rendering the hair coupe mesh with Kajiya & Kay's illumination model – (Kajiya, et al., 1989) – and shadows are added by using Deep Opacity Maps – (Yuksel, et al., 2008).

The completion of this project has resulted in a fully configurable hair simulation & rendering framework that is able to display physical and visual believable human hair.

Preface

The work described in this thesis was carried out between October 2009 and June 2010 and is the result of the master thesis of the study Human Media Interaction followed at the University of Twente, Enschede.

For nine months I have been working on the project of building a framework that is able to simulate and render virtual human hair coupes. Working on this thesis has been a satisfying experience that has provided me with many opportunities to use my knowledge gained during the studies – Computer Science (CS) and Human Media Interaction (HMI) – to solve the various problems encountered during the development of the framework. The thesis has also proven also very valuable for broadening my knowledge in the field of Computer Graphics and Human Simulation, as some knowledge can only be gained in practice. I hope that – with the completion of this thesis – I have made a useful contribution to the field of Graphics & Virtual Reality.

I would like to take this opportunity to thank all of the people that helped me during the years to come to the point that I can receive my academic diploma by completing the Final Project of Human Media Interaction:

First of all I want to thank my graduation committee: Dr. Job Zwiers, Dr. Mannes Poel and Ir. Herwin van Welbergen for their feedback and support during the Final Project and their proofreading of my master thesis. I also want to thank all other teachers from whom I followed courses; they have extended my knowledge about Computer Science and Human Media Interaction and provided me with the academic skills required to finish an academic study. Last, but not least, my special thanks goes out to my friends and family who have supported me over all these years and provided me with a comfortable environment where I could dedicate a great deal of my attention towards my study.

The paper currently lying before you is the result of my dedicated work and my curiosity into the subject. I hope that your interest in reading the master thesis will be of the same amount as my interest of writing the master thesis.

Mike Lansink,
Enschede, July 2010

1. Introduction

In the fields of Computer Graphics and Human Simulation, hair offers one of the most challenging physics and rendering problems and has therefore been one of the least satisfactory aspects of rendered human images to date. In recent years, virtual characters are increasingly present in our environment, whether in video games, animated films or advertisements. Correctly modelling the shape and appearance of the hair is crucial if one wants to generate realistic looking characters, but it also raises many problems due to the intrinsic properties of hair.

A human hair is, geometrically, a long, thin, curved cylinder. Typically, there can be from 100,000 to 150,000 individual hairs on a human head. Depending on the individual, the hairs can vary in width from 45 µm to 100 µm. The strands of hair can have any degree of waviness from perfectly straight to kinky. The hair colour itself can vary from white through grey, yellow, red, and brown, to black, depending on the degree of pigmentation.

When viewed as an aggregate material, consisting of many strands, hair exhibits subtle optical properties which affect its appearance. The specular highlights in particular, owing to the geometry and orientation of the individual strands, are highly anisotropic. Furthermore, the interaction of light between the strands of hair is highly complex. Each strand reflects light on adjacent hairs and casts shadows. In backlighting situations, a significant amount of light is also diffusely transmitted through the hair itself, which is partially translucent. Finally, the hair width is small enough to cause, in some situations, noticeable diffraction effects.

All of this makes the problem of finding an accurate hair illumination model an extremely difficult one, as attempting to explicitly simulating and render the individual cylindrical surfaces of each hair is clearly impractical. The overwhelming complexity of hair makes it very difficult to create a believable computer model of it. It comes to accurately simulate the shape and movement of hair, and realistically handling complex optical phenomena, such as for example dispersion of light, self-shadowing, occurring within a hair.

In this project I explored existing physics and illumination models for rendering believable human hair. A selection from these promising methods, models and techniques are incorporated into a hair simulation & rendering framework, which covers the following functionalities:

- **Hair creation** (selection of hair type and hairstyle)
- **Hair physics** (influence of gravity, elasticity and collision)
- **Hair shading** (presence of lighting, shadowing and colouring)

In the project, I have looked for opportunities to improve the framework. These improvements include: simplifying methods to increase performance, optimizing shaders, algorithms and other techniques, and eliminating bottlenecks in the rendering pipeline by tuning the CPU feeds for the graphics and the vertex and pixel pipelines, which include finding the right balance of workload distribution between the CPU and GPU.

The project has resulted in a working hair simulation & rendering framework that is able to display physical and visual believable human hair.

1.1.Project Goal

As mentioned in the introduction, the main goal of this research project is to create a framework that allows the simulation and rendering of physical and visual believable human hair.

Researchers developed several models and techniques to tackle the different sub problems encountered in the domain of hair simulation and rendering. When comparing the different solutions for these problems, it becomes clear that within almost every situation no single method, model or technique can be called the optimal solution to that specific situation. The choice of implementation always revolves around a trade-off between: simplicity, performance and quality.

The framework created in this project provides a solution to the main problem of hair simulation by solving the different sub problems found in hair rendering and simulation, and combining them with each other. In doing so, the selection of models and techniques used for solving the sub problems need to be compatible with each other. Furthermore the implemented application should find a good overall balance between simplicity, performance and quality.

The project has resulted in a working hair simulation & rendering framework which is able to display physical and visual believable human hair. A user of the framework is able to specify the hair and environment properties, and the framework will compute and generate the visual representation of the hair coupe.

1.2.Overview of the Thesis

In the following chapters I will describe the techniques, which I have implemented into the framework. Here follows a general overview of the chapters in this thesis:

I will start with the chapter titled ***Natural Hair Properties***, in which I will give a description of the physical, mechanical and optical properties of natural hair.

The next chapter, ***Related Works***, contains an overview of all relevant information related to digital hair simulations. This chapter will list the work of earlier attempts in simulating and rendering hair.

The chapter ***Hair Simulation*** contains a comprehensive explanation of the developed hair simulation & rendering framework; I will motivate the design decisions that have been made and explain the implemented models and techniques used. This chapter will end with a description of the results that can be achieved with the framework.

Finally, the chapter ***Conclusion*** will have a comparison between the developed hair framework and the formulated research goal, and will end with a discussion and evaluation of the project.

The last sections are appendices – ranging from ***Appendix A*** to ***Appendix D*** – containing additional information of the project in the form of:

- **Implementation** of the shader code used for rendering head and hair
- **Examples** of the parameter input files used for physics calculations
- **Screenshots** from hair coupes taken from within the framework
- **Listing** of the of the hardware and software used in this project

2. Natural Hair Properties

To get a better understanding of the different hair simulation models and techniques created throughout the years – which will be explained later in this thesis – one first needs to get a good understanding of the properties and behaviour of natural human hair. This chapter is not intended to describe in detail all the physical, biological and chemical properties of hair, but will only focus on the properties which are interested for the simulation and rendering of static or dynamic human hair.

I will starts with two definition of the word hair. According to the online dictionary ***Dictonary.com*** (http://dictionary.reference.com), the word hair is defined as such:

Hair /hɛər/ -**noun**:

1. *any of the numerous fine, usually cylindrical, keratinous filaments growing from the skin of humans and animals; a pilus.*

2. *an aggregate of such filaments, as that covering the human head or forming the coat of most mammals.*

From the definition listed above it becomes clear that the word hair can refer to two things. It can be seen as an individual hair strand or it can be seen as a collection of strands which forms a complete person's hair coupe. Both, individual hair strands and hair as an aggregate material, have remarkable properties and behaviours important for simulation and rendering. These properties and behaviours will be explained in the following sub sections.

2.1.Individual Hair Strands

This section contains a description of the properties and behaviour of an individual hair strand. Important topics for individual hair strands are: the natural hair shape, the layers found in a hair cross section, the mechanical properties of individual hairs, and optical properties of individual hairs.

2.1.1. Hair Shape

A human hair fibre is a thin filament structure (less than 0.1 mm in diameter) and has a circular or oval cross section. The fibre can be divided into two distinct parts. The first part, the active part of the hair, called the hair follicle, is located approximate 4 millimetres under the skin and is responsible for the production of keratin, which is a structural protein responsible for the stability and shape of the hair material. The production of keratin takes place in the bulb, which is the area of proliferation of the hair follicle. The second part, the visible – and dead – part of the hair, is called the hair shaft, and corresponds to the hair strand. The hair strand is entirely synthesized by the associated follicle, which acts as a mold for shaping the strand.

The shape of the strand depends largely on the thickness of the sheaths surrounding the hair follicle. A thicker external root sheath would result in a greater pressure on the formation of the hair shaft; the shaft, somewhat rigid and malleable at this early stage of construction, would adopt an elliptical shape, which it retains once the keratin stiffen and lose their biological activity.

The formation of the hair shaft thus can be seen as an extrusion from the follicle, and its shape remains relatively constant along the entire strand. This means that the hair strand has an almost uniform cross section, natural twist and natural curvatures all along the shaft. These geometric

parameters are responsible for the shape of the hair and will determine whether the hair becomes straight, curly, or fuzzy.

The geometric properties of hair are relatively continuous defined between the different people on the planet, with the hair shape ranging over naturally straight, wavy, curly, fuzzy and kinky, and the cross section of the – more or less elliptical – ranging from 45 to 100 µm. While the geometric properties of hair are relatively continuous defined between different people, they can nevertheless be used for schematically distinguishing between three broad categories of hair in the world, namely: African, Asian and Caucasian.

African people have follicles with a helical form and an oval cross section, whereas Asian people have follicles that are almost completely straight with a larger and circular cross section. As a result, Asian hair is thicker, with no natural curliness. It makes it look smooth and regular. In contrast, African hair looks frizzy, curly and irregular. Caucasian people have hair that is the intermediary between the two previous mentioned categories. Present mainly in the offspring of European ancestry (Europe, North America, Australia, etc.), Caucasian hare may have a shape very variable, ranging from steep to curly.

The longitudinal section of the hair follicle and hair strand can be seen in Figure 1. This illustration shows that the hair shaft – or hair strand – is created in the hair bulb of the hair follicle and exits the skin at the epidermis, which is the outermost layer of the human skin. The internal and external root sheaths form the shape of the cross section.

(Bertails, 2006) (Robbins, 2002)

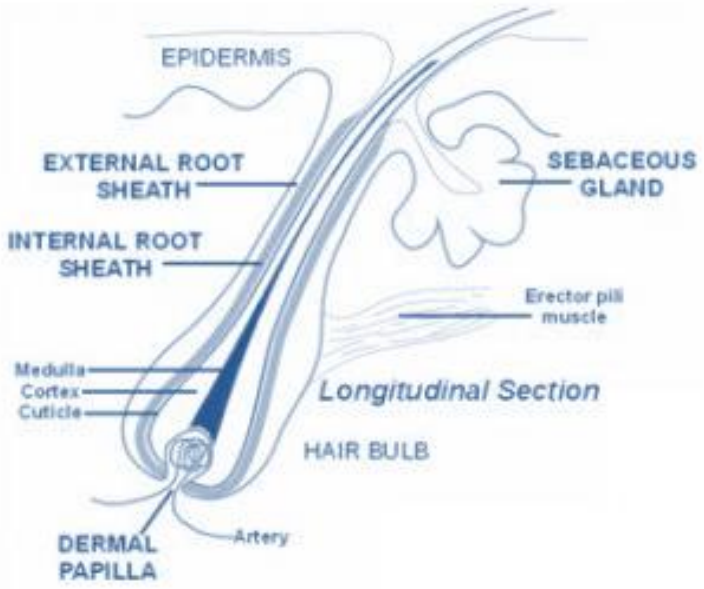

Figure 1. Longitudinal Section of Hair

2.1.2. Hair Cross Section

A hair strand can be divided into different layers. The cross section of a hair strand is roughly composed of three concentric layers, which are from the outside in:

- The cuticle; the protective covering of hair, composed of non-pigmented cells, also called horny scales. These scales are flat rectangular cells arranged on the hair shaft like the tiles on a roof and form multiple overlays. The free margin of the scale is oriented toward the tip of the hair.
- The cortex; the structural body of the fibre, representing 90% of the total hair weight. Composed of cells filled with keratin, the cortex provides the physical and mechanical properties of the fibre. The cortex contains also grains of melanin; the pigment responsible for the colouring of the hair.
- The medulla; a central canal in the core of the strand, which consists of degradation products of the cortex cells, airspaces and fats. The medullary canal has a honeycomb structure and is largely responsible for the thermoregulation properties of hair from mammals. Humans however largely lost this thermal property.

The cross section of a hair strand can be seen in Figure 2. This illustration shows the three layers that make up the hair strand, from the outside in: cuticle, cortex, and medulla. The inner and external root sheaths that are responsible for the shape of the cross section are also visible.

(Bertails, 2006) (Robbins, 2002)

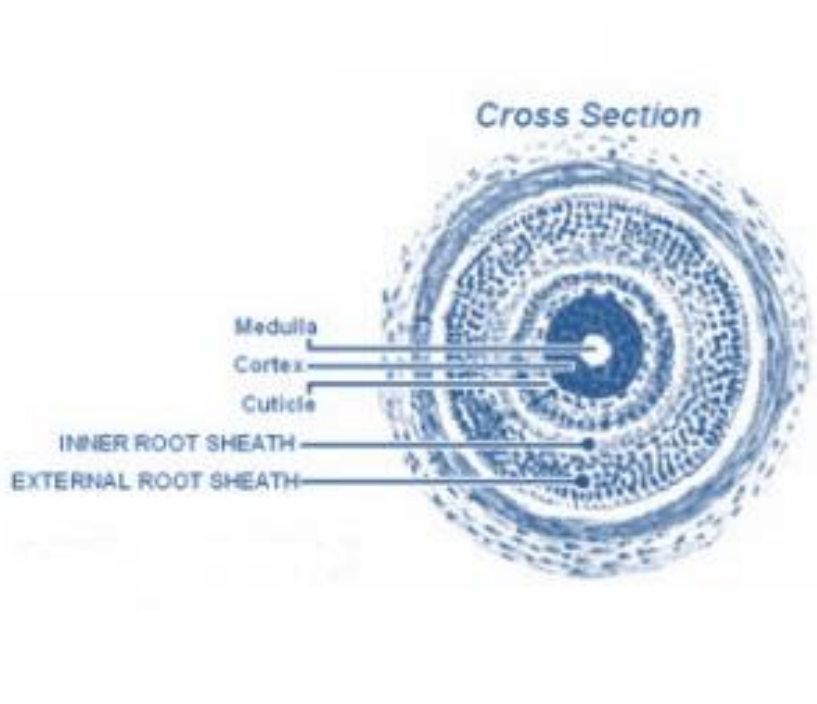

Figure 2. Cross Section of Hair

2.1.3. Hair Mechanical Properties

The mechanical properties of a hair stand are determined by the cortex. The cortex consists mainly of keratin, which is a remarkable stiff material, making the shaft extremely difficult to shear and stretch. However, because the cross section of the hair is very small, the strand can be easily bent and twisted; hair, a very rigid material, remains inextensible during movement, but the thin cross section, will give it significant bending and torsion deformation. The thin structures are elastic: after motion,

they tend to come back to a rest shape, which is related to their individual natural curliness and to the set of external forces applied to them.

When a thin hair wisp is held between two hands, and the hands are brought closer to each other, the wisp will react by bending in a direction perpendicular to the applied compression. If the hands are brought even closer together, a second instability occurs and the strand suddenly starts to coil (the bending deformation is converted into a twist).

In bending, one can notice that a hair reaches a limit beyond which it can no longer bend. This happens due to the instable behaviour of hair: it abruptly leaves the plane of the flexion and twists onto itself. This happens because at a certain level of restraint imposed by the bending, the configuration of the hair in pure bending has a potential energy greater than the torsional configuration, which "reduces" the hair to a more natural position and "relieves" the hair from the strain it experiences. This behaviour also explains why the natural curl of a hair will never lie in one single plane, and follow more or less the form of a ringlet (tight ringlets result in curly hair, while loose ringlets result in wavy hair). Note that this experiment on the bending of the hair can be achieved by oneself, quite simply, by using a single hair or a thin wisp.

An illustration of the above mentioned experiment is shown in Figure 3. This illustration shows the various instabilities of a hair when the ends of a hair wisp are brought closer together. **A**) shows the hair at rest, **B**) shows the buckling effect by moving both ends of the hair together, and **C**) shows that the buckling of the wisp converts into a twist when the limit of bending has been reached.

(Bertails, 2006) (Robbins, 2002)

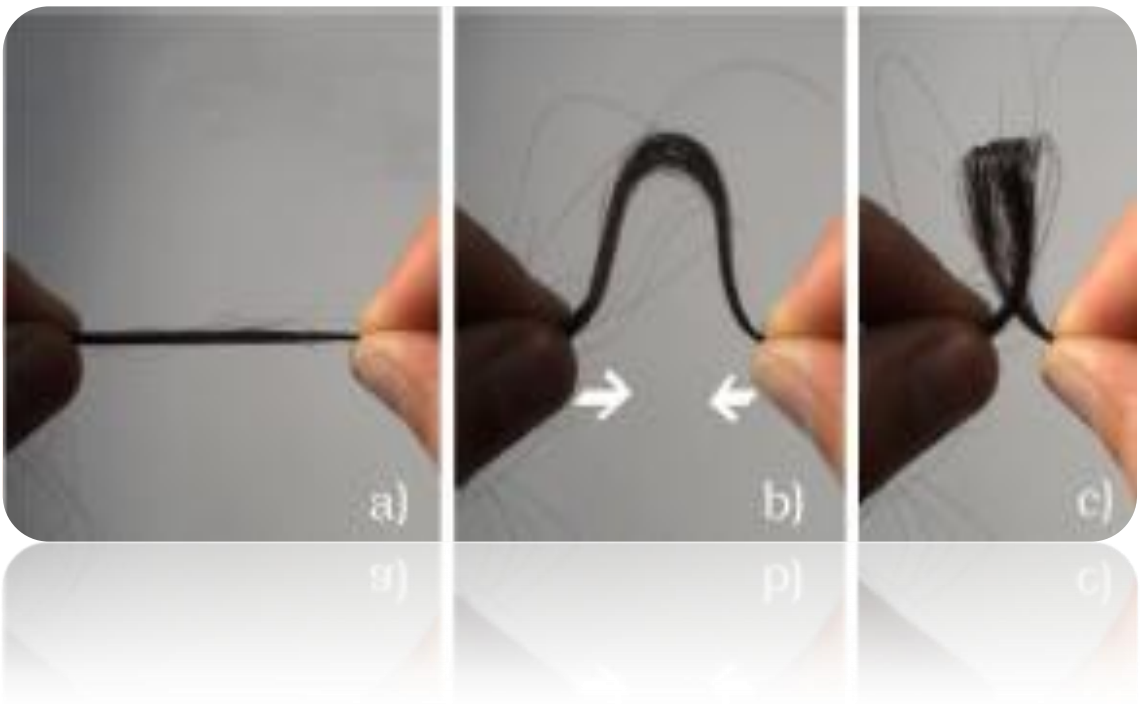

Figure 3. Hair Bending and Twisting

2.1.4. Hair Optical Properties

The optical properties of an individual hair strand are primarily determined by the translucent material of the hair, the amount of pigmentation the hair contains, and the geometric properties of the surface of the hair strand.

A hair fibre is a translucent material with a constant refractive index of 1.55. As explained earlier, the cortex of the hair fibre contains grains of melanin, which are responsible for the colouring of the hair. There exist two types of melanin: the most common type is eumelanin, which is responsible for the black or brown colouring of the hair and the less common type is pheomelanin, which colours the hair yellow or red (depending on the amount of pigment).

As with human skin, the colour of the hair is determined by the melanin. The darker eumelanin and lighter pheomelanin are distributed in different proportions in the human hair, which result in the different colour tones of hair among different people. In science a grouping of 11 to 12 different colour tones is used, ranging from black to light blond. The natural appearance of a person's hair is genetically determined and depends on the skin tone and eye colour of that person. In grey hair, the hair shaft does not store pigment; the production of melanin has subsided and the melanin is replaced with deposited air bubbles.

The special structure of the surface of the hair, composed of overlapping scales, has a major influence on how the hair interacts with light. As a semi-transparent cylinder, a portion of the light is reflected directly on the fibre surface, which causes a specular reflection with the same colour as the light. Another portion penetrates the fibre by refraction, is absorbed partially, and then returns through subsurface scattering to the outside, which causes a diffuse reflection with the same colour as the pigments within the fibre. But unlike a perfect cylinder, the presence of scales on the surface of the fibre deviate the light rays slightly by an amount of approximately 3 degrees. This deviation will cause the phenomenon of dispersion.

The optical properties and behaviour of hair can be captured and mathematically explained by a simplified model. The model contains a transmission component, and three reflective components, which include specular reflection, internal reflection (after passage of the rays in the fibre), and diffuse reflection (rays that are absorbed by the fibre and re-emit the light on the same side as the light source is located).

Because of the scales, specular reflection and internal reflection follow different directions, and produce separate peaks of reflection. The specular reflection is the primary hair reflection and has the same colour as the light, the internal reflection is the secondary hair reflection and adapts to the colour of the pigments present within the hair.

An illustration of above mentioned model is shown in Figure 4. The illustration shows the specular reflection (R, Reflection), and the internal reflection (TRT, Transmission/Reflection/Transmission). The reflections are deviated by a small angle of 2α due to the scales tilted by an angle of α

In this illustration it becomes clear that the diffuse reflection increases when the hair is less pigmented. Pigments in hair absorb light refracted through the hair, and thus the more a hair is pigmented, the more the light is absorbed. Strongly pigmented hair fibres absorb more light than lesser pigmented hair fibres, which re-emit the light received in a diffuse form.

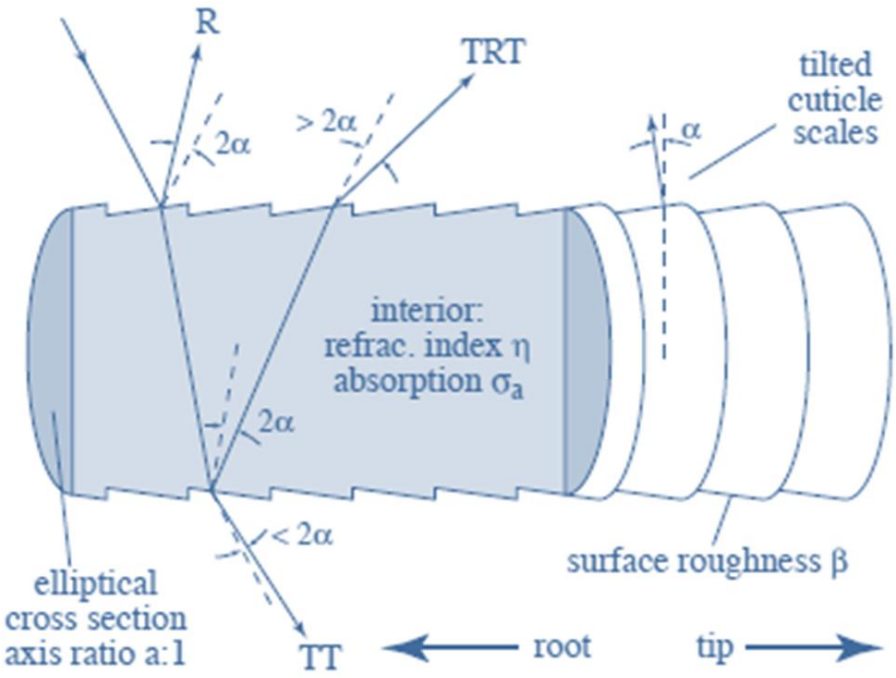

Figure 4. Marschner Illumination Model

Another property which can be observed from Figure 5 is that it is possible for light to pass through the hair strand (TT, Transmission/Transmission). Lesser pigmented hair, such as blond, tends to let more light pas then darker hair, which results in a backlighting of the hair when the light source is placed behind the hair, viewed from the angle of the observer.

(Marschner, et al., 2003), (Bertails, 2006)

Figure 5. Backlighting of Blond Hair

2.2. Full Hair Coupe

This section contains a description of the properties and behaviour of a full human hairstyle. Important topics are: hair distribution, hair motion, and the optical properties of hair.

2.2.1. Hair Distribution

Human hair is composed on average of 100 000 to 150 000 hairs implanted alone or in groups of two or three in the scalp. The total number of hair on the head depends on several factors: the diameter and colour of hair, ethnicity, age of the person and the genetic predisposition whether one loses his hair prematurely. The capillary density (average 200 to 300 hairs per square centimetre) is variable from one person to another, but also with a region of the scalp to another in the same individual (in general, the capillary density is strongest on the top of the head and decreases near the temples).

In the Caucasian group of people with straight and short hair, it is common to observe patterns formed by the hair. These capillaries are caused due to the fact that, unlike the Asian hair that grows right on the head, Caucasian hair grows extremely sideways to the scalp. In the case of short and smooth hair, level and angle of implantation of hair determines the direction of the hair on the head. Sometimes the implantation directions can vary quite strongly on a single scalp, which result in capillaries patterns appearing; note that African hair is too curly to be able to present such behaviours. In brushed and groomed hair there often appear spiral patterns at a central point near the crown of the head. In general, these spiral patterns rotate in clockwise direction.

(Bertails, 2006)

2.2.2. Hair Motion

As mentioned in previous section, hair is composed on average of 100 000 to 150 000 hair strands. All these strands interact with each other and with the body. Hair contacts have a complex nature due to the tilted scales that pave the cuticle of the hair shaft. The irregular surface causes anisotropic friction forces, whose amplitude strongly depends on the direction of sliding with respect to the orientation of the scales. This can be felt when rubbing a fibre between thumb and forefinger. Due to the lightness of a strand, these friction forces are much higher than other external forces. The friction between hair strands explains the strong cohesion observed in hair motion.

Global hair motion and even the shape that hair takes at rest highly depend on the nature of the multiple interactions taking place between hair strands: collisions and contacts between hair strands of different orientations cause hair to occupy a pretty high volume when considering the size and radius of individual hair strands.

The type and shape of strands have a significant effect on collective hair behaviour. The most obvious example of this is hair curliness. Hair clumps are more likely to appear in curly hair, where the greater number of contacts among hair strands increases the probability that they will become entangled. In contrast, strands of straight hair slide easily with their neighbours, and so they appear as a continuum rather than as a set of disjoint wisps. In general, as the complexity of hair geometry increases, there is more internal friction and fewer available degrees of freedom for motion. Due to the larger volume, tangled or fuzzy hair in motion is much more subject to air damping than smooth and disciplined hair.

(Cani, et al., 2006) (Robbins, 2002)

2.2.3. Hair Optical Properties

The optical properties of a full hairstyle are primarily determined by light scattering on the surfaces of the individual hair fibres. Particularly for light-coloured hair, where light rays can deeply penetrate the hair, the sub-surface scattering of those rays contributes for a large fraction to the overall colour.

Individual human hair fibres have very distinctive scattering properties. Light coloured fibres can scatter a large portion of the light into the hemisphere that is facing away from the light source, and the scattered light is confined to a fairly sharply defined cone of directions with inclinations near that of the incident direction of the light ray. In a strongly forward scattering material a large contribution comes from multiple scattering compared to single scattering, since the majority of the incoming light is scattered into the material where it cannot be directly observed. The multiply scattered light both contributes to the aggregate reflectivity of the hair and causes a spatial spreading of the light, softening geometric features and blurring the hard shadow edges.

Scattering in hair is different from scattering in more isotropic material, however. Unlike materials with no oriented structure, in which only the scattering angle matters, scattering in hair depends strongly on orientation. Light in hair is scattered much more broadly across the strands than along them. The overall direction of the light rays remains constant, even after several scattering events.

A comparison of some global light properties seen in light and dark-coloured hair can be seen in Figure 6. Light-coloured hair is stronger forward scattering then dark-coloured hair, causing the light to spread out through the hair and blurring the shadow edges.

(Marschner, et al., 2003) (Moon, et al., 2006)

Figure 6. Global Light Properties in Hair

3. Related Work

This chapter contains a general overview of all the information relevant to this project. It will give an introduction to previous work done in the domain of digital hair simulation and rendering.

As mentioned by (Rosenblum, et al., 1991), the simulation of virtual hair can be divided into three distinct problems: Hair modelling, Hair animation, and Hair rendering

Hair modelling can be viewed as modelling the shape of the hair at rest, incorporating the geometry of the hair and specifying the density, distribution, and orientation of the hair strands. Hair animation involves the dynamic motion of hair, including collision detection between the hair and objects, such as head, body or mutual hair interactions. Finally, the hair rendering entails colour, shadows, light scattering effects, transparency, and anti-aliasing issues related to the visual appearance of the hair on the screen.

Recently, modelling, animating and rendering hair represent very active areas of research in the field of Computer Graphics. In the following sub sections I will present previous works of researchers and explain the models and techniques developed over the last decades that have been used for tackling the problems found with digital hair creation.

3.1.Hair Modelling

This section contains a description of previous works of researchers solving the problems found in digital hair modelling. Important topics are: the layout and growth of hair, and the modelling and styling of hair.

3.1.1. Hair Layout & Growth

Given the large number of hair strands composing human hair, it is unimaginable to manually specify the location of every hair on the head of a virtual character. To make this task feasible in a reasonable time, a number of two and three-dimensional techniques have been developed for determining the hair layout. To simplify the complexity of the hair, some methods use the fact that hairs that are packed close together near the head often have similar forms throughout their length: these hair strands are grouped together into wisps.

Some of the modelling approaches do not place the hair directly in 3D on the head of a virtual character, but offer the possibility to paint the location of the follicles of hair on a 2D texture map, which is then projected onto the head in 3D. In the approach of (Rosenblum, et al., 1991), this map is called the *follicles map*, and it allows the user to not only treat half of the scalp laid flat on a texture, but also handles the other half automatically by using the symmetrical properties of a head. The final mapping to the head is made by a 3D spherical projection. This method is also used in (Yu, 2001).

In the method of (Kim, et al., 2002), the user can use predefined geometric shapes to interactively select areas on a 2D parametric surface. The selected area on the 2D surface is then prepared on the 3D scalp by the manipulation of control points, and the sections of strands drawn on the surface then serve as building blocks for the hair on the mesh.

An illustration of (Kim, et al., 2002) method is shown in Figure 7. In this illustration the 2D parametric map is placed on the 3D head by manipulating control points. Selecting a circular area on the parametric map will draw a circular section of hair on the head.

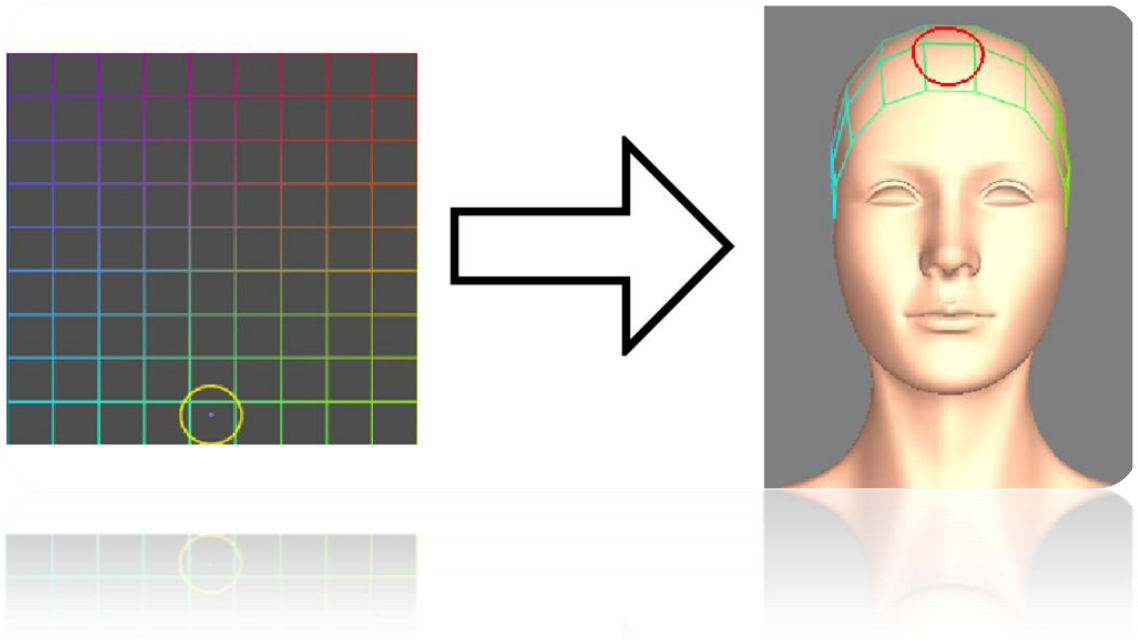

Figure 7. Hair Surface Selection

Another alternative is to directly set the roots of hair on the 3D scalp. (Patrick, et al., 2004) have proposed an interactive interface through which the user can select the point on the 3D triangular mesh of the head on which the user wishes to implement hair; the area on the scalp is specified by selecting triangles. Each triangle selected in this approach forms the basis of a hair wisp, which means that the geometry of the strands strongly depends on the mesh of the virtual head. This can cause problems, particularly when one wants to control the shape of a wisp.

Once the placing of follicles is specified, one needs to consider how the individual hairs should be placed in this area. Most approaches use a uniform distribution of follicles on the scalp, which corresponds fairly well to reality, at least in the case of a normal hair. Typically, a wisp is then defined by a 3D curve, which serves as its skeleton, and an outline with a simple shape (e.g. a circle or a triangle) will define an envelope around the skeleton. Individual hair strands are then placed inside this envelope, often using a 2D uniform distribution as its cross section, starting at the roots. This approach is used by (Plante, et al., 2001), (Chen, et al., 1999), and others. However, if the envelopes at the roots of the different wisps overlap, parts of the scalp will have a greater density of hair, which can cause visual artefacts.

To ensure a uniform distribution of hair on the head, (Kim, et al., 2002) begin by evenly distributing the follicles of the hair on the scalp, and then assigning one (and only one) wisp to the overlapping area. For density variations, some approaches allow the user to paint the density levels on the scalp; an example is (Baker, et al., 2003). These densities are mostly displayed in 3D by using a colour coding. Controlling the density parameter can be useful for synthesizing special cuts where the hair is shaved in places. Colour codes can also be used to determine the shape and length of a hair wisp.

3.1.2. Hair Modelling & Styling

The modelling and styling of hair can be divided into three categories: methods based on geometry only try to model believable hair shapes by modifying the geometry, methods based on physics try to find a physical analogy of hair and determine the shape of hair by physical formulas and algorithms,

and methods based on images try to extract the hair shape from an image or pictures. The hair modelling techniques developed are explained in the following sections.

3.1.2.1. Methods based on Geometry

Most of the modelling techniques that are used for generating hairstyles are geometrical in nature. They are typically based on a parametric representation of the hair, which can be changed by the user to change the corresponding hairstyle. The advantage of such an implementation is that it offers great flexibility. The geometric primitives often used are: parametric curves when dealing with individual hairs or parts of the skeleton, and parametric surfaces for the representation of 2D wisps. In general these representations are only used to define a global shape for the hair, while details, such as ringlets and buckles, are added procedural, by using for example sinusoidal modulation of the primitives or by changing the shape at a different level of detail.

3.1.2.1.1. Overall Shape

Many approaches choose to represent the wisps of hair by parametric surfaces, such as (Koh, et al., 2000), (Koh, et al., 2001), (Liang, et al., 2003), (Noble, et al., 2004) and (Scheuermann, 2004). NURBS[1] are often used as the parametric surface. These methods are relatively inexpensive in computing time, as the use of the pieces of parametric surfaces (also called hair bands) to represent large groups of hair greatly reduces the complexity of the hair, while providing maximum flexibility for modelling the shape. The appearance of the individual hairs can be expressed through the use of a texture applied to the surface.

With the intention to generate non-photorealistic results, (Mao, et al., 2004) has developed a technique to make cartoon-style hair strip The method also offers an interface based on the outline of a sketch to quickly build a hairstyle: given a 3D head model, the user interactively specifies the area where the hairs should be implanted into the head, and draws in the front view a silhouette of the desired hairstyle. The system then automatically generates a surface representing the contour of the hair; hair bands are then created between the surface and the head.

A major drawback of the 2D modelling of wisps is that the created hairstyles are usually quite unrealistic, flat, and lack the notion of volume. To tackle this problem, (Liang, et al., 2003) extrudes the hair band by adding fins to the side of the band. These fins are obtained by projecting each point of the hair band onto the scalp and linking two by two the points projected to the origin. The U-shape obtained for each band can then make the hair more voluminous.

(Kim, et al., 2000) have developed a specific method to increase the visual realism of hairstyles modelled from parametric surfaces by forming a set of thin shells, called *thin shell volume*. This method generates thickness for the hair by making offset surfaces that are aligned parallel to the initial surface. The individual hairs are then distributed within the volume and constructed, and their shape is modified with an algorithm that mimics the effect of a comb. The hair generates a volume, and the surface exhibits realistic hair patterns.

(Noble, et al., 2004) also use NURBS surfaces to model and animate real-time cartoon-style hair, but their approach is very different. Here, a single NURBS surface is used initially to model the envelope of the volume occupied by hair. This surface can be animated by controlling key positions, and will

[1] Non-Uniform Rational B-Splines

define the basic movement of the hair. The animation of small individual strands near the surface will add detail to the global movement.

An illustration of a model that uses parametric surfaces to model hair is shown in Figure 8. The hairstyle is created by using several patches of NURBS which are overlapping each other. This model is very simple and cheap for simulation and rendering.

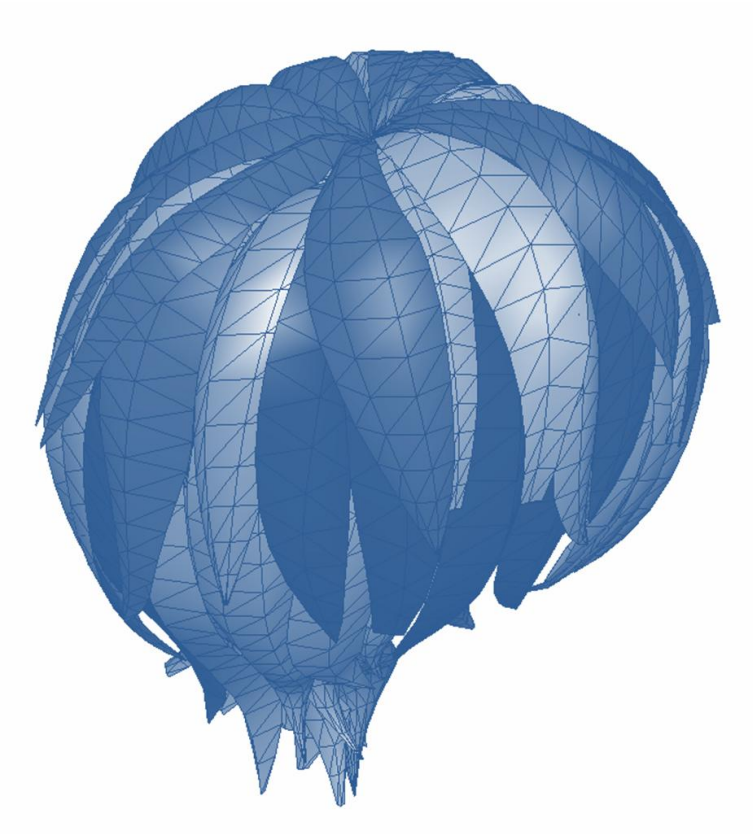

Figure 8. NURBS Surfaces

Other approaches model a wisp as a trigonal prism with three B-spline curves, which are used to define its envelope. Papers that uses a trigonal prism are (Watanabe, et al., 1991) and (Chen, et al., 1999). The wisp obtained has the shape of a generalized prism with three sides and has the advantage of being a 3D object, while being defined by a limited number of parameters. To draw individual hairs within the wisp, 2D coordinates are used as barycentric coordinates in the triangular sections of the wisp. The coordinates are randomly selected and define the positions of the interpolated hair strands confined within the trigonal prism.

Finally, several methods use a generalized cylinder model for each wisp of hair. Some examples are (Plante, et al., 2001), (Kim, et al., 2002), and (Patrick, et al., 2004). A generalized cylinder is defined by a parametric curve, which serves as its skeleton, and a function defining the radius of the hair wisp along the skeleton. Various forms of wisps can be easily obtained by interactive manipulation of the skeleton and by adjusting the radius parameter. In general, the modelling of individual strands by generalized cylinders is useful to create hairstyles where the hair strands are very visible, or where the hair shape is determined by constraints such as braids or ponytails.

3.1.2.1.2. Detailed Shape

Once the overall shape of the hair is determined – using a geometric approach or a physical model – it could be desirable to add details to the hair such as waves or ringlets, to increase the natural realism of the hairstyle.

(Yu, 2001) generates different types of hair curling by modulating the positions of the hair by a trigonometric function, which can be easily customized. Through these functions, the user can control the frequency of loops produced and their amplitude. It is also possible to add noise levels to the groups of hair; noticeable irregularities within the hair coupe will increase the realism of the hair.

In the method of (Choe, et al., 2005), the hair is modelled as wisps, and the overall shape of each strand is determined by the shape of the skeleton, which are called the guide strand. The shape of the hair strand can be decomposed into two levels: a global shape, piecewise linear, which gives the general shape of the skeleton, and added details. These details are constructed from a hair prototype by using a Markov chain in which the degree of similarity between the guide strand and the created prototype are controlled through a Gibbs distribution. The degree of self-similarity of hair is controlled by several random parameters, such as length or distance to the master strand, and by adding noise to the geometry. The hairstyles are globally consistent with the shape of the original guide strands, but show noticeable variations at a more detailed level.

3.1.2.1.3. Multi-Resolution

(Kim, et al., 2002) and (Kelly, et al., 2003) represent a hair through a complex hierarchy of cylinders, allowing the user to select the level of detail with which he forms the shape. The coarse levels of detail can quickly create a global shape, while detailed levels allow for more direct control over the fine details of geometry. Complex hairstyles can be created quickly by operations such as copying and pasting an already shaped wisp to other strands.

3.1.2.2. Methods based on Physics

The geometric approach is well suited for creating complex and varied hairstyles. However with this type of approach, the user must fully model the hair by hand, which could be long and tedious work. Moreover, if such hairstyles are designed to be animated with key positions – limiting the degrees of freedom of the hair to the degrees of freedom of the key positions – they will become not suitable for modelling physical correct dynamics. Identifying the physical parameters is not trivial in this situation.

While approaching the generation of hairstyles by physical models is less flexible and less easily controlled by the artists, they can have a number of advantages: First, they are partially automating the modelling phase, next, the model generated is physical correct, and finally, they can more easily predict the influence of any external factor (moisture, cosmetics, airflow) on the shape of the hairstyle.

Three important physical based methods that will be explained in the following subsections are static cantilever beams, vector fields and elastic Cosserat rods. All these methods lack proper hair-hair collision handling which result in a hairstyle that lacks in volume. Two solutions to tackle this problem are described in the last sub section.

3.1.2.2.1. Static Cantilever Beams

A classical case study about materials strength is the statics of a cantilever beam which has one of its two edges embedded in a support, and the other end free of restraints. (Anjyo, et al., 1992) was inspired by this work to generate the shape of a single hair which was has its root embedded into the scalp. Considering that the only external force applied is gravity, they use a very simplified linear equation derived from the theory of static cantilever beams, defining in each point of the hair the bending angle that causes equilibrium between gravity and elasticity. Because the equation is linear - and theoretically valid for only very small deformations - this method does not properly summarize the extrusion of hair under the influence of gravity. In particular, in this approach, a vertically implanted hair extrudes straight up, without falling over, and it must constantly add more strength to the system (artificial forces, which are called *styling forces*) for a realistic falling of hair. By combining this technique with pseudo-physical management of collision between the hair and the body, the ability to add other external forces to disrupt the shape of the hair, and a cutting tool, the method manages to generate several smooth hairstyles.

(Lee, et al., 2000) have used the same physical model to generate the overall shape of the hair, but extended to handle dynamics and adds a trigonometric function that can procedurally add details in the form of curls and ringlets.

Note however that the cutting operation of both approaches is not physically correct; the cutting does not correspond to the removal of material, it is simply done during the render phase. The shape of the hair at the top of the head stays the same, whatever the level is where the hair is cut, contrary to what happens to real hair.

3.1.2.2.2. Vector Fields

Other physical approaches to model hairstyles are almost all based on the definition of a vector field in space to generate the shape of the hair. Examples can be found in the papers (Yu, 2001) and (Choe, et al., 2005). The vector fields can be two-dimensional or three-dimensional. These methods extend the concept of *styling forces* already used in the methods presented in the previous section.

To generate complex hairstyles, while offering the user an interactive interface with a maximum control over the shape of the hairstyle, (Yu, 2001) proposes the use of static 3D vector fields. The user can simply generate a vector field by selecting vector field primitives form an available library. Each of these fields, mainly from the domain of electromagnetism, distorts the hair (repulsion, winding around an axis, creating ripples). By combining these vector field primitives and assigning them each a local influence on the hair, it is possible to create a global field in 3D, which represents the direction of hair in space. Each hair is then automatically drawn from the root as a series of elements, following step by step the 3D direction indicated by the global field, until the hair reaches the desired length.

(Choe, et al., 2005) also uses 3D vector fields, but uses the elasticity of natural hair to create hairstyles. Each hair is modelled as a sequence of elements, and for each hair strand the algorithm computes via energy minimization the angles between the right segments that represent the best of both the influence of global vector field and the tendency of hair to regain their natural shape at rest. Another important feature of this method is the ability for users to define constraints for modelling the hair. Specifying a constraint produces a local vector field constrain, which will be applied to the overall vector field. The same energy minimization algorithm is then applied to calculate the

deformation of the hair in the presence of a constraint. In practice, the user can specify three types of constraints: constraint points, trajectory constraints and directional constraints. The constraints can be used to create complex hairstyles such as braids and ponytails.

An illustration of (Yu, 2001) vector field method is shown in Figure 9. In this illustration the hairstyle is displayed together with the directions of the vector fields that are responsible for the shape of the hairstyle. The different colours of the vectors represent the different vector fields that influence the hair.

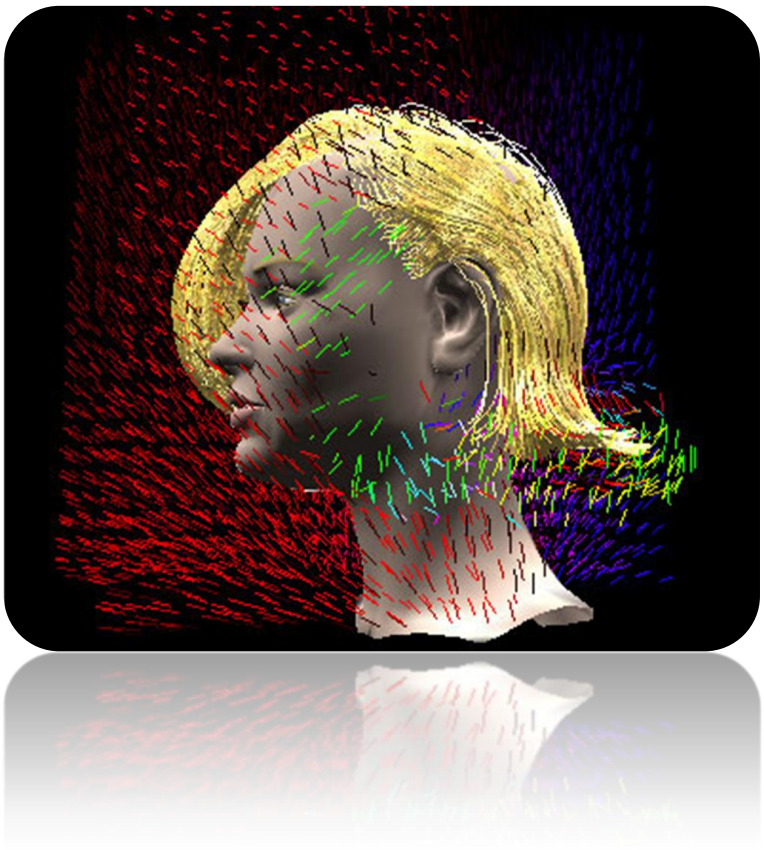

Figure 9. Vector Fields

3.1.2.2.3. Static Cosserat Rods

One of the newest methods, based on physics, is developed by (Bertails, et al., 2005b). This paper uses the Kirchhoff equations for static Cosserat Rods to calculate the shape of a hair strand. This method is based upon a mechanically accurate model for static elastic rods, which accounts for the natural curliness of hair, as well as for the hair ellipticity.

The model of the Cosserat rod is a mathematical representation of a rod material. In this context, a rod material can be explained as a slender object whose dimensions in the two directions of the plane of the cross section are negligible compared to the longitudinal dimension. Hair is thus a good example of a rod material.

The Kirchhoff equations describe the deformation of these elastic inextensible rods. These equations, although made in the context of Hooke's linear law elasticity, are nonlinear and can describe the changes in form from the natural configuration of a rod. The Kirchhoff equations take into account all modes of deformation observed in real hair (bending, torsion and longitudinal inextensibility). In

addition, all parameters involved in these equations are related to simple properties such as the hair natural curvatures, hair ellipticity and hair bending stiffness.

(Bertails, et al., 2005b) calculates the shape of the hair by applying gravity and collisions to the Cosserat rods. The final hair structure is calculated by computing the shape for which the hair contains the minimum amount of potential energy. The potential energy is the sum of elastic energy, gravitational energy and collision energy – where collision energy is a penalty energy added when a strand penetrates an object.

Although the model is complex, it yields various typical hair configurations that can be observed in the real world, such as ringlets. The method is able to account for the structural properties of the different hair types existing in the world – Asian, African and Caucasian hair – by setting very few input parameters. This method is also able to simulate the non-linear instabilities in real hair, which are mentioned earlier in section ***Mechanical Properties***, located in chapter ***Natural Hair Properties***.

3.1.2.2.4. Creating Volume

While most geometric methods give intrinsically volume to the hair – the use of primitive geometric modelling in a volume – it is not the same for the physical based models, which must explicitly address the interactions between hair (modelled directly on the forces applied) to generate hair with a natural volume.

Collision detection between hairs is a very expensive operation, and therefore incompatible with the constraint imposed by an interactive hairstyles modelling system, (Lee, et al., 2000) has developed a technique that adds volume to hair without worrying about the exact collision occurring between individual hair strands. The principle is based on the following hypothesis: the hair growing on the top of the head should always cover the hair growing below. The idea then is to create a series of envelopes around the head, and detecting the collision of each hair with the envelope corresponding to the elevation of the hair strand. This method is simply the extension of the algorithm for detecting collisions between the hair and the head. Of course, it only works if the head remains static and oriented vertically.

(Choe, et al., 2005) considers the density of hair in space as an indicator of collisions; when the density of a region of space reaches a certain threshold, a collision is detected, and hair in that region of space is then forced to occupy a greater volume.

3.1.2.3. Methods based on Images

The methods for creating hair with physical models partially automate the creation phase of hair, and thus significantly speed up the process. However, most of these methods are unable to automatically generate the geometry for fine and detailed hair, and therefore use procedural methods for adding details geometrically. To automate the creation of 3D hairstyle, researchers have recently proposed a new type of approach based on the reconstruction of hair from photographs.

(Kong, et al., 1997) were the first to use photographs of hair to automate the 3D reconstruction. However, their approach only allows reconstructing – from different viewpoints – the shape that the volume of hair occupies, while hair strands are generated heuristically within the volume. The method does not guarantee any resemblance between the orientations of the generated hair and that of natural hair

(Grabli, et al., 2002) proposed a new idea. They analyze the reflection of light on hair of the taken picture to infer the local orientation of the hair in 3D. For this, they work with one point of view, but use a mobile controlled light source, which allows them to generate multiple images of a single hair lit from different directions. These images are then analyzed and filtered to determine the 3D orientation of hair that is most plausible. But this technique restores the hair only partially. (Paris, et al., 2004) and (Wei, et al., 2005) are extending this approach to generate a more complete reconstruction. In this method, multiple viewpoints are considered, and the strategy is to test multiple oriented filters per pixel, where the orientation given by the local filter shall be taken as most reliable in this specific location. This technique allows reconstructing the entire visible part of the hair.

3.2.Hair Animation

This section contains a description of previous works of researchers solving the problems found in digital hair animation. Different approaches taken to animate hair are: the animation of individual hair strands, the animation of a continuous medium and the animation of collection of disjointed wisps. The last section will cover multi-resolution methods that combine methods from the three previously mentioned approaches.

3.2.1. Individual Hair Strands

Four types of models have been proposed in computer graphics to simulate the dynamics of individual hair. These are: the spring-mass system, projective angular momentum, the articulated rigid-body chain, and Static Cosserat Rod. Each of these models is described in the following subsections.

3.2.1.1. Mass-Spring System

(Rosenblum, et al., 1991) presented one of the first methods to animate individual hair. In this approach, each hair is represented as a chain of particles connected by stiff longitudinal springs and angular springs called hinges. Each 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. This model is simple and easy to implement, however does not take into account some important characteristics of hair, such as torsional forces. Because of the very stiff springs, which are needed to retain the inextensibility property of hair, this method becomes numerically unstable and can only simulate by taking small time steps. The implementation was only able to simulate about 1000 hair strands.

Other fields of research in computer graphics, such as cloth simulation, proved that the integration of implicit representations are well adapted to stiff problems, because they guarantee unconditional stability of the system, even for larger time steps. Some methods for simulating hair were later adopting implicit representations for calculating the movement of mass-spring chains, such as (Ward, et al., 2003), (Sokol, 2003) and (Selle, et al., 2008). Other methods, such as (Plante, et al., 2001), simulated curls that allowed the longitudinal springs to be stretched slightly, which resulted in less stiff equations and numerical calculations that are more stable.

3.2.1.2. Projective Angular Moment

A well known approach to simulate the dynamics of individual hair strands has been proposed by (Anjyo, et al., 1992) which is based on the theory of cantilever beams.

Originally, during the modelling phase of the hair, the rest form of the hair is given by a simplified equation of the static cantilever beams. Each hair is then regarded as a chain of rigid elements, and its dynamic movement is calculated as follows:

- Each element s_i is treated as a direction in space, and can be parameterized by spherical coordinates ϕ (azimuth) and θ (zenith).
- The external force F applied on the element (e.g. weight) is projected onto the plane and P_ϕ on the planeP_θ, respectively defined by the angles ϕ andθ. The longitudinal projection of force is being neglected, since it is not intended to affect the rigid element.
- The fundamental principle of dynamics is then applied to each of the two degrees of freedom ϕ and θ element, which gives two scalar differential equations of order 2, which are solved numerically at each time step. This method always calculates the configuration of the elements making up a hair from root to tip.
- Collisions between the hair strands and head are simply treated as a pseudo-force field acting from within the head. This force field is being modelled as an ellipsoid.

Figure 10 shows an illustration of the spherical coordinates of a hair elements_i. Node p_{i-1} and p_i form the connection to the previous and next element of the hair strand respectively. ϕ (Azimuth) contains the horizontal angle of the element located in the x-z plane while θ (Zenith) contains the vertical located in the ϕ-z plane. For each element s_i, ϕ and θ of that element are taken as input. And gravity and pseudo-force fields acting on element s_i will perturb ϕ and θ at s_{i+1} to represent the orientation of the next element s_{i+1}. Linking these elements will result in the cantilever hair strand.

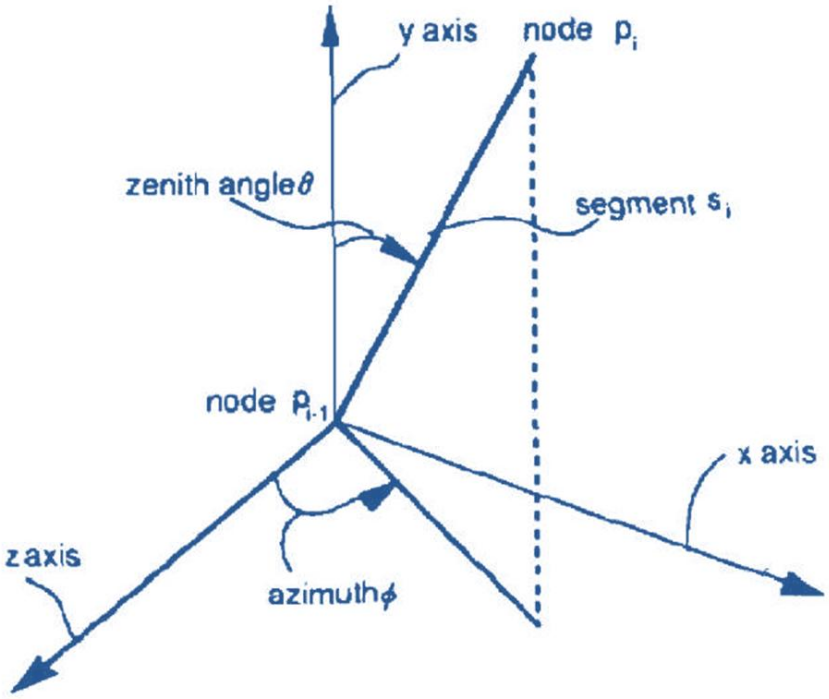

Figure 10. Spherical Coordinates of a Hair Element

(Anjyo, et al., 1992) method has many advantages: it is easy to implement, stable, efficient (tens of thousands of hairs can be computational simulated within reasonable time), and it produces good results on straight hair (the inextensibility condition of the hair is intrinsic to the model, and bending rigidity is modelled convincingly).

Many subsequent papers have implemented this dynamic model. To give some examples: (Kurihara, et al., 1993), (Daldegan, et al., 1993), (Lee, et al., 2000), and (Ward, et al., 2003) all use Projective Angular Moment to calculate the shape of the individual hairs. However, because this model simplifies the physics of hair so much (e.g., it does not model the phenomenon of twisting the hair or the influence of the lower part of the hair on the top), it will sometimes generate unrealistic behaviour of hair, especially in the case when handling collisions.

3.2.1.3. Articulated Rigid-Body Chain

Papers such as (Hadap, et al., 2001) and (Chang, et al., 2002) have facilitated the dynamics of articulated chains to simulate individual hair strands. Being more complex than the previous approach, this approach also more accurately models the physics of hair, meaning that the internal stresses to the hair are treated bilaterally (upper hair has an impact on the bottom, and vice versa). Moreover, the articulated chain systems are well known in the field of robotics, and efficient algorithms have been developed for some time to simulate the dynamics of such systems in linear time.

(Hadap, et al., 2000) proposed a hybrid model of dynamic hair, taking into account both the individual behaviour of hair and complexity of interactions occurring between hairs. The hair is generally regarded as a continuous medium for interaction management while some hair conditioners are simulated to increase the visual realism. Each hair is represented with an open articulated chain, parameterized by generalized coordinates to keep only the degrees of freedom corresponding to the curvature and torsion (stretching and compression are prohibited). Therefore, this amounts to model each hair with a set of rigid elements connected by bal joints, each with three degrees of freedom. In addition to weight, two types of forces act on each element: first, a force which reflects the rigidity of the element bending and torsion, and second, a force in response to interactions that occur in the hair (hair collisions, collisions between the hair and body, and interaction between the hair and air), calculated by simulation of the continuous medium.

The system dynamics is simulated by the algorithm with linear complexity in number of elements. (Chang, et al., 2002) use a similar approach to animate guide hair.

3.2.1.4. Dynamic Cosserat Rods

In (Bertails, et al., 2006), the Static Cosserat Rods model, mentioned earlier in this paper, is extended to incorporate dynamics. The energy minimization algorithm which is used in (Bertails, et al., 2005b) is replaced by Lagrangian mechanics. This results in a physically correct and stable dynamics, capable of simulating hair with large intermediate time steps.

The Cosserat Rod, which represents the shape of the hair, can be reconstructed by recursively adding new element to the end of the hair strand. Every element is represented by three parameters, which are: torsion and bending in the two dimensions of the cross section of the element. When taking these parameters as input for the Lagrangian calculations, they serve as generalized coordinates which can be used to calculate the coordinates for the next time step. The newly acquired

coordinates are then used to reconstruct the element for the new time step. The calculation of the coordinates requires solving a dense inertia matrix, which results in an algorithm with quadratic complexity in the number of elements composing the hair.

(Bertails, 2009) proposes a new, recursive scheme for the dynamics of a Cosserat Rod, inspired by the Featherstone algorithm for Articulated Multi-Body Chains for recursively solving the motion of the hair strand. The key of the approach relies on two features: first, the recursive nature of the kinematics and second, the observation that accelerations of the elements are linear functions of the applied force. Therefore, cumulate inertias and force for the linked elements can be pre-computed within a first pass by propagating internal joint forces from the free end to the fixed end of the hair strand. Finally, the forward dynamics is recursively solved within a second pass traversing the hair strand in the reverse way. The linear complexity of the method makes it possible to allow more elements to be used in the representation of a hair strand.

3.2.2. Full Hair Coupe

As it is often not possible to calculate an entire hair coupe in real-time, models are developed that, by making assumptions, reduce the complexity of calculations.

Three types of models have been proposed in computer graphics to simulate the dynamics of a full hair coupe. These are: continuous medium, disjointed wisps, and the multi-resolution approach. Each of these models is described in the following subsections.

3.2.2.1. Continuous Medium

Some papers assume that hair can be viewed as having an overall cohesion in the movement, allowing them to model the hair as a continuous medium, which can result in a significant reduction of the complexity of the system to animate. These papers will be represented in the following sub sections.

3.2.2.1.1. Fluid Medium

(Hadap, et al., 2001) try to manage the complexity of hair-hair interactions, hair-air interaction and hair-body interaction by treating all of the hair as a continuous medium governed by the laws of fluid dynamics. This model is composed of a limited number of individual hairs – simulated as articulated chains – immersed in a continuous medium represented by Smoothed-Particle Hydrodynamics[2] (SPH), and subject to the laws of fluid dynamics.

The interactions between individual hairs and the surrounding fluid are used to model the interactions that occur within the hair. The full motion of the hair therefore comprises a global movement, governed by fluid dynamics, and motion at a finer scale of a few individual hairs to increase the visual realism of the animation. The result produced is quite convincing, since the volume of the hair is properly maintained during animation.

The disadvantage of this method is that it leads to very smooth hair animations, which does not display the discontinuities that one observes in the real case. Recently, (Bando, et al., 2003) were inspired by the idea of a continuous medium for the hair, and extending to an adaptive model.

[2] Smoothed-particle hydrodynamics is a computational method used for simulating fluid flows

3.2.2.1.2. Guide Hairs

Considering that hair has an overall coherence in its appearance during movement, one can simply animate only a few guide hairs, then expand this movement to the rest of the hair by interpolation. This technique has been proposed by (Daldegan, et al., 1993). The disadvantage is that it only applies to very smooth hair, and, also, shows no discontinuity in the movement. In addition, this method poses the problem of collisions between the hair and body, since only a few hairs are able to detect interpenetration.

(Chang, et al., 2002) use strips of triangles connecting the guiding hair to compensate for this disadvantage. Managing collisions between hairs is then simplified by only treating the interactions that occur between the guide hairs and these intermediate triangles, which is inexpensive. In addition, guide hairs are automatically added adaptively to ensure that all guide hairs are properly distributed during movement.

3.2.2.2. Disjointed Wisps

Another idea is to consider that hair has not an overall continuity, but rather has local internal cohesion. The hair is then modelled as a set of geometric primitives called wisps, which are animated separately. Obviously, this model also reduces the number of individual elements to animate, while keeping some complexity to the hair. Several studies have been based on this approach, some simulating a small number of coarse wisps, others using surfaces for wisps, such as (Koh, et al., 2001) and (Koh, et al., 2000), or volumes for wisps, such as (Noble, et al., 2004), these wisps can be deformable or rigid.

To address the complex interactions occurring between hairs during major moves, (Plante, et al., 2001) introduced a layered wisp model. In this model collisions between wisps are processed in two different ways, depending on the orientation of strands involved:

- If the wisps are almost parallel, the method considers that there is no real collision, but rather a continuous movement. The interwoven strands are subject to forces of viscous friction.
- If the wisps have different orientations, the method considers that there is a collision with a discontinuity in the movement. Both wisps repel each other and are very dissipative, meaning that they will not bounce from each other.

However all these methods are far from interactive. For example, simulations conducted by (Plante, et al., 2001) required a calculation time of about 40 seconds per image.

3.2.2.3. Multi-Resolution

Previous described methods that take into account interactions between hairs are far from interactive. Recently, adaptive methods have been developed to minimize the computing time of simulation while maintaining maximum realism.

(Ward, et al., 2003) models a hair using three levels of geometric detail: 2D hair bands are used at the most coarse hair group, 3D hair wisps are used at the intermediate level, and individual hairs are used at the highest end. The transitions between these levels of detail are managed dynamically during the motion according to certain visual criteria, such as camera position, for example. This method is effective, especially during the rendering step.

(Bando, et al., 2003) has modelled the hair by a set of Hydrodynamic Smoothed-Particles interacting with each other, each adapting interactions during movement. In this model, a particle represents a certain amount of hair provided with an orientation (the orientation of a particle is the average orientation of all the hair represented by the particle). Initially, the model construct chains of connection between neighbouring particles aligned along the length of hair; two neighbouring particles with similar directions, such that their positions are aligned along this direction are connected by a spring. This initial configuration is maintained throughout the movement, since it reflects the spatial coherence of interactions between particles.

During movement, each particle is more likely to interact with particles belonging to his current neighbourhood (different from the original neighbourhood). The method proposes to manage these dynamic interactions between hair by establishing close links between particles, the links disappear when two particles are nearer to each other. The method facilitates the separation and grouping of hair cutting, while maintaining a constant length for the hair. At each time step, the search neighbourhood of each particle is performed efficiently through a grid of voxels.

3.3.Hair Rendering

This section contains a description of previous works of researchers solving the problems found in digital hair rendering. Important topics are: explicit and implicit hair representations, hair Illumination models and hair shadowing techniques.

3.3.1. Hair Representation

As mentioned in previous sections, 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. For example, the explicit models involve rendering algorithms that take geometric primitives such as elements or triangles as input, while volumetric models can only use algorithms that can manage their implicit representation. How researchers in the domain of virtual hair rendering handle the explicit and implicit representations of hair will be described in the following two subsections.

3.3.1.1. Explicit Representation

When using an explicit representation for the hair – where each individual hair will be represented individually – an individual hair is naturally represented by a curved cylinder. Early works of hair rendering, carried out by (Watanabe, et al., 1991) provided a representation of hair by using trigonal prisms. In this representation, each strand of hair is seen as a series of prisms linked by three faces. Because each face of the prism is assigned a unique colour, this method assumes that the colour variation over the hair thickness is negligible. 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 caused by the very special geometric nature of the hair (hair is extremely thin - less than 0.1 mm in diameter on average) may occur. Under normal viewing, the thickness of a hair projected on the screen is much smaller than the size of a pixel. This property is important because problems can occur when using under-sampling algorithms for rendering the polygonal geometry. All rendering algorithms based on geometry explicitly determine the colour of a pixel (or depth) based on a limited number of discrete samples; sub sampling creates abrupt changes of colour, or noisy edges around the hair. By increasing the number of primitives used, the problem reduces, but also reduces the speed of the algorithms.

(LeBlanc, et al., 1991) have solved this problem, by finding the appropriate mix of colours for the pixels, by using *blender buffer* techniques. In this method, each hair is drawn as a broken line and the shaded colour is mixed within a pixel buffer. The order of drawing is carefully observed, showing first the polygons near the camera and finishing with the polygon furthest from the camera. (Kim, et al., 2002) also used an approximation of visibility sorting, to allow hair with a transparency value.

3.3.1.2. Implicit Representation

Volumetric textures (or texels), first used in (Kajiya, et al., 1989), avoid the problem of aliasing by using pre-filtered shading features. These pre-filtered shading features are then stored in basic primitives, called volumetric cells (or voxels). The voxels can have different dimensions, depending on which level of detail is required for the rendered image.

The final rendering of the image can be performed by tracing rays from the camera, and through the voxels. All voxels on a ray's path will contribute to the final colour of the pixel. The methods based on ray tracing often generate good quality results, and the cost of ray tracing through the volume are relatively reasonable for short hair or fur, but can become very heavy and expensive in the case of longer hair. Moreover, if the hair is animated, the voxels features and attributes within the volume should be updated and recalculated at every time step, which reduces the benefits of pre-initial screening significantly.

3.3.2. Hair Illumination

To make a human hair look realistic, it is necessary to take into account both the optical properties of each hair fibre, 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 fibres are illuminated, while global hair properties also include the way in how the fibres cast shadows on each other; self-shadowing particular important for creating a volumetric appearance of hair.

The first sub section will review the different models that have been proposed to reflect the local optical properties of natural human hair. The second section will describe the different models that are used for creating shadows within hair rendered by local illumination models. A third section is included, describing the global illumination models which are meanly used for off-line hair rendering.

3.3.2.1. Local Illumination Models

The two most popular illumination models used in hair rendering are Kajiya & Kay, which is introduced in (Kajiya, et al., 1989), and Marschner, introduced in (Marschner, et al., 2003). These two models will be explained in greater detail in the following sub sections.

3.3.2.1.1. Kajiya & Kay

The first and most popular model of light scattering by a hair is (Kajiya, et al., 1989), which was originally designed for the rendering of a teddy bear's fur. This model is characterized by the sum of two components; a diffuse component and a specular component.

The illumination model follows directly from the underlying cylindrical nature of a human hair strand. Figure 11 shows an element of hair and the important vector quantities used in the model. The unit tangent vector ($\boldsymbol{t}$), represents the direction of the hair's axis. The light vector ($\boldsymbol{l}$) points in the direction of the light source. The reflection vector ($\boldsymbol{r}$) points in the direction of reflected light and is the direction of maximum specular reflection. The eye vector ($\boldsymbol{e}$) points in the direction of the

observer. The angles θ and ϕ are the angles from the tangent vector to the light vector and from the tangent vector to the eye vectors respectively.

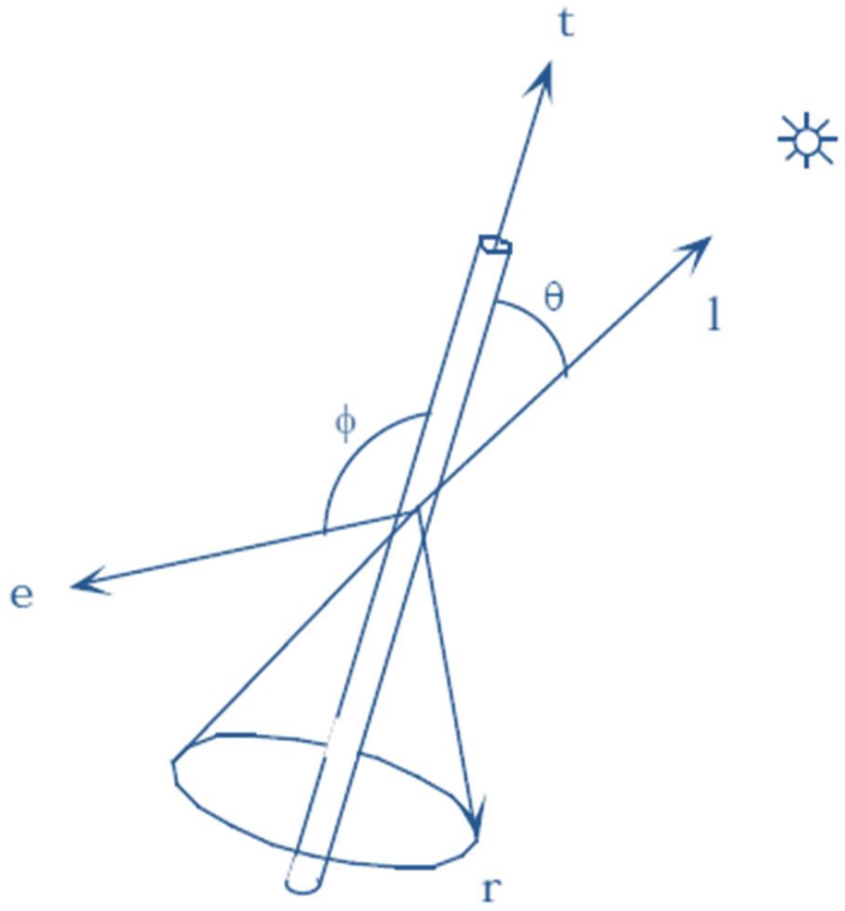

Figure 11. Kajiya & Kay Illumination Model

The diffuse component of the reflectance can be obtained by integrating the Lambert cosine law along the illuminated half of the cylinder to yield a simple function of the angle θ. The diffuse term can be calculated with:

$$I_{diffuse} = K_d \sin\theta$$

Where K_d is the coefficient of diffuse reflectance. This function does not take into account self-shadowing, that is, the fact that half of the hair strand is in shadow. Therefore, when the hair would be an opaque cylinder, the diffuse component would vary with what visible fraction is in shadow, which is dependent on the direction of the eye vector $\boldsymbol{e}$. However, human hairs are actually quite translucent so that the area in shadow transmits diffusely an amount of light comparable to the amount reflected diffusely on the non-shadow side of the hair. As a result, this simple formula makes an approximation that works very well in practice.

The specular component is derived essentially by taking a Phong specular distribution and rotating it around the hair axis. This argument can be motivated by considering symmetry. At any section along the length of the hair, there are surface normals pointing in all directions perpendicular to the tangent vector $\boldsymbol{t}$. Therefore, the 180 degree semicircular surface of the hair, which is not in shadow,

will directly reflect light in a cone-shaped 360 degree radial pattern formed by rotating the $\boldsymbol{r}$ vector around the hair axis.

This cone represents the angles of maximum specular reflection. The cylindrical Phong specular component is calculated by taking the angle between this cone and the eye vector $\boldsymbol{e}$, which equals $(\cos\theta\cos\phi + \sin\theta\sin\phi)^n$, and using this angle for the standard Phong equation:

$$I_{specular} = K_s(\cos\theta\cos\phi + \sin\theta\sin\phi)^p$$

The diffuse and specular components, together with an ambient component result in a hair illumination equation of:

$$I_{KajiyaKay} = L_a K_a + \sum_{lights} \left(L_i(K_d \sin\theta + K_s(\cos\theta\cos\phi + \sin\theta\sin\phi)^p)\right)$$

Where L_i is the light intensity, K_a is the ambient reflectance coefficient, and L_a is the ambient light intensity which is present in the environment.

3.3.2.1.2. Marschner

(Marschner, et al., 2003) has proposed a more complete and more natural model for local hair illumination. Their model improves the approach of Kajiya & Kay on two essential points. First, it predicts the variation of the reflection of light rays by the rotation of ϕ around the axis of the fibre, taking into account the elliptical cross section of the hair. Second, it models the effect of the scales of the hair on the specular and internal reflection of light, and therefore obtains two specular peaks, which can also be seen in actual measurements of real hair.

(Marschner, et al., 2003) has also contributed to a better understanding of real natural hair. The Marschner model treats each individual hair fibre as a translucent cylinder and considers three possible paths that light may take through the hair. These three paths are also displayed in Figure 4, and labelled R, TT and TRT. (Marschner, et al., 2003) showed that each of these paths contributes to a distinct and visually significant aspect of hair reflectance, allowing a degree of realism not possible with previous hair reflectance models.

While the contributions of the different individual paths are rather complex to calculate, the model can be summarized as the sum of the colours of the individual paths:

$$I_{Marchner} = Path_R + Path_{TT} + Path_{TRT}$$

Where $Path_R$, $Path_{TT}$ and $Path_{TRT}$ correspond to the light contributions of the primary specular reflection, diffuse subsurface reflection and secondary specular reflection respectively.

3.3.2.2. Shadowing Models

The phenomenon of shadows and hair self-shadowing contribute very much to the appearance of hair volume. Figure 12 illustrates this; the left image shows hair rendered with shadows, and the right image shows hair rendered without shadows. Hair rendered with proper shadows appears to have more depth and look much more realistic.

Figure 12. Hair Shadowing

3.3.2.2.1. Deep Shadow Map

To handle self-shadowing of translucent objects, (Lokovic, et al., 2000) proposed an extension to the traditional shadow maps: maps of deep shade (deep shadow maps). For each pixel of the map, the algorithm stores a function of transmittance (also called visibility function), which gives the proportion of light arriving at each depth z sampled along a ray launched from a light's point of view and a list of fragments is stored for each pixel.

The transmittance function defined by these fragments is then compressed into a piecewise linear function of approximate transmittance. The value of the transmittance function starts decreasing after the depth value of the first fragment in the corresponding light direction. The shadow value at any point is found similar to shadow maps, but this time the depth value is used to compute the transmittance function at the corresponding pixel of the deep shadow map, which is then converted to a shadow value.

3.3.2.2.2. Opacity Shadow Map

(Kim, et al., 2002) have proposed a practical way to implement the deep shadow maps, they have rendered maps of opacity (opacity shadow maps), and have applied it to the problem of hair shadowing. (Ward, et al., 2003) have implemented the same technique and combined it with the representation of hair at different levels of detail, thus aiming at optimizing the animation and rendering of hair, but did not get a framework that could simulate at an interactive level.

The technique of opacity maps has also been exploited by other authors, such as (Mertens, et al., 2004) and(Koster, et al., 2004), who have taken advantage of new functionality within graphic cards to further accelerate this method, and achieved real-time interactive applications.

Opacity shadow maps can be seen as a simpler version of deep shadow maps, and are designed for interactive hair rendering. It first computes a number of planes that slice the hair volume into layers. These planes are perpendicular to the light direction and are identified by their distances from the

light source (i.e. depth value). The opacity map is then computed by rendering the hair structure from the light's view. A separate rendering pass is performed for each slice by clipping the hair geometry against the separating planes.

The hair density for each pixel of the opacity map is computed using additional blending on graphics hardware. The slices are rendered in order starting from the slice nearest to the light source, and the value of the previous slice is accumulated to the next one. Once all the layers are rendered, this opacity map can be used to find the transmittance from the occlusion value at any point using linear interpolation of the occlusion values at the neighbouring slices.

Depending on the number of layers used, the quality of opacity shadow maps can be much lower than deep shadow maps, since the interpolation of the opacities between layers generates layering artefacts on the hair. These artefacts remain visible unless a large number of layers are used.

Figure 13 shows an illustration of the opacity shadow map. The hair volume is sliced into different layers by using the clipping planes and the densities are stored into opacity maps. These opacity maps are used during the final rendering to calculate how much light penetrates to the corresponding layer and thus how much of the hair is in shadow.

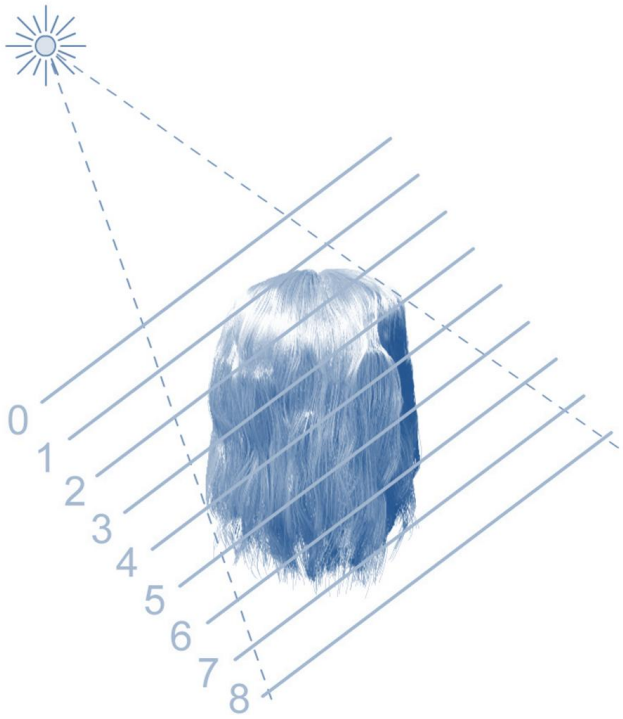

Figure 13 Opacity Maps

3.3.2.2.3. Deep Opacity Maps

(Yuksel, et al., 2008) have proposed an alternative to the opacity map, which is called deep opacity maps. Deep opacity maps use a algorithm that use two passes to prepare the shadow information, and use an additional pass to render the final images with shadows.

The first step prepares the separation between the opacity layers. A depth map (shadow map) is rendered as seen from the light source, which gives a per pixel depth value Z_0 at which the hair geometry begins. Starting from this depth value, the hair volume is divided into K layers such that

each layer lies further from the light source than the previous layer. Because the algorithm uses a shadow map to determine where the K layers are positioned, the layers will follow the shape of the hair structure.

The second step renders the opacity map using the depth map computed in the previous step. This requires that the hair needs to be rendered only once. As each hair is rendered, the Z_0 value from the depth map is used to find the depth values of each layer on the fly. The opacity contribution of a hair fragment is added to the corresponding layer in which the hair fragment falls and all layers behind it. The total opacity of a layer at a pixel is the sum of all the contributing fragments.

The opacity map can be represented by associating each colour channel with a different layer, and accumulate the opacities by using additive blending on the graphics hardware. One channel is reserved for storing the depth value, so that it can be stored in the same texture with the opacities. Thus, by using a single colour value with four channels, three opacity layers can be stored. Multiple texture lookups can be used when one wants to use more opacity layers during final rendering.

One disadvantage of using a small number of layers with deep opacity maps is that it can be more difficult to ensure all points in the hair volume are assigned to a layer. In particular, points beyond the end of the last layer do not correspond to any layer. Mapping these points onto the last layer, will usually give good results. Unlike opacity shadow maps, opacity interpolation occurs within the hair volume, thus hiding possible layering artefacts.

Figure 14 shows an illustration of the deep opacity shadow map. The hair volume is sliced into different layers by using a previous calculated shadow map and the densities are stored together with the shadow map into the different colour channels of the deep opacity map. The deep opacity map is used during the final rendering to calculate how much light penetrates to the corresponding layer and thus how much of the hair is in shadow.

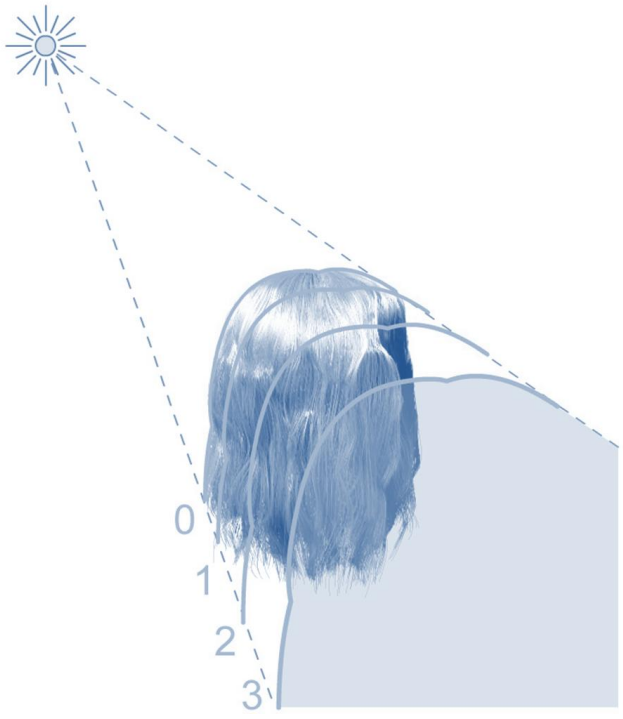

Figure 14. Deep Opacity Maps

3.3.2.3. Global Illumination Models

Physically based simulation of light scattering in hair can reproduce the appearance of real hair accurately. (Marschner, et al., 2003) shows that hair fibres transmit a large fraction of the light that falls on them and that, because of this, multiple scattering is important to the appearance of hair – in fact, according to (Zinke, et al., 2004) and (Moon, et al., 2006), multiply scattered light predominates in blond and other light coloured hair with low absorption values. An accurate hair renderer must therefore allow for inter-reflections among the fibres, accounting for all paths by which light can be scattered from fibre to fibre before reaching the eye. However, simulating global illumination in a model consisting of tens of thousands of densely packed fibres is a complex problem.

Global Illumination Models can roughly be divided into two broad categories. These are Path Tracing and Photon Mapping. Each of these models is described in the following sub-sections.

3.3.2.3.1. Path Tracing

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. Path tracing is a computer graphics rendering technique that attempts to simulate the physical behaviour of light as closely as possible. It is a generalisation of conventional ray tracing, tracing rays from the virtual camera through several bounces on or through objects. This technique is the simplest, most physically-accurate and slowest rendering method. It naturally simulates many effects that have to be specifically added to other methods, such as soft shadows, depth of field, motion blur, caustics, ambient occlusion, and indirect lighting. In order to get high quality images from path tracing, a very large number of rays need to be traced lest the image have lots of visible artefacts in the form of noise.

(Kajiya, et al., 1989), (Gupta, et al., 2005), and (Yuksel, et al., 2007) are three examples of papers that use Path Tracing for illuminating the hair.

3.3.2.3.2. Photon Mapping

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. In computer graphics, photon mapping is a two-pass global illumination algorithm that solves the rendering equation. Rays from the light source and rays from the camera are traced independently until some termination criterion is met, and then they are connected in a second step to produce a radiance value. It is used to realistically simulate the interaction of light with different objects. Specifically, it is capable of simulating the refraction of light through a transparent substance such as glass or water, diffuse inter-reflection between illuminated objects and the subsurface scattering of light in translucent materials.

(Moon, et al., 2006) and (Zinke, et al., 2008) are two examples of papers that use Photon Mapping for illuminating the hair. (Moon, et al., 2008) improved this technique for hair rendering by storing the position- and direction-dependent scattered radiance distribution in a 3D grid of coefficients. Compared to a photon map, this representation is more compact and better organized in memory and outperforms previous implementation of photon mapping with a factor of 20 while achieving equivalent quality.

4. Hair Simulation & Rendering Framework

This chapter contains a comprehensive description of the hair simulation & rendering framework created in this project. The first section is about the design, and describes the models and techniques that are incorporated into the framework. This section will also give a motivation why these models and techniques are chosen over the alternatives. The second section lists the results that can be acquired with the application. This section will investigate the performance of the application, and looks at the appearance of the rendered virtual hair image.

4.1.Design

As mentioned earlier in ***Related Work***, the simulation of virtual hair can be divided into three distinct problems: Hair modelling, Hair animation, and Hair rendering.

Hair modelling can be viewed as modelling the shape of the hair at rest, incorporating the geometry of the hair and specifying the density, distribution, and orientation of the hair strands. Hair animation involves the dynamic motion of hair, including collision detection between the hair and another object – where the other object can be a head, a body or another hair strand. And hair rendering entails the colouring, shadows, light scattering effects, transparency, and anti-aliasing issues related to the visual appearance of the virtual hair rendered to the screen.

The initial approach in this project was to incorporate all of the above mentioned aspects of virtual hair simulation into the framework. However, early attempts to understand and implement the Dynamic Cosserat Rod model of (Bertails, et al., 2006) into the framework have proven to be too difficult to build within the given time constraints of this project. My knowledge about physics and Lagrangian mechanics, required to implement the dynamics of the Cosserat model, are inadequate for tackling this problem. Furthermore, (Bertails, et al., 2006) skips some crucial information in their paper, required to replicate their Dynamic Cosserat Rods.

Instead, I have chosen for an earlier version of the Cosserat Rod model, which is presented in (Bertails, et al., 2005b). This model determines the configuration of a hair strands by performing an energy minimization on the hair strand. The potential energy of the hair strand is the sum of the elastic and gravitational energy stored within the rod and the penalty forces added due to hair penetrating geometry in the scene. The configuration for which the potential energy is the minimum represents the static rest shape of the strand. The Static Cosserat model however cannot account for the dynamic movement of a hair strand. Therefore I have chosen to exclude the problem of hair animation from this project, allowing me to concentrate on hair modelling and hair rendering.

The reason that I want to use the Cosserat Rod model to determine the shape of the hair strands, is that it is the only physical hair model, known to date, that can correctly represent the non-linear elastic behaviour of natural hair – such as the bending and twisting instability— which is required to create waves, curls and ringlets often seen in the different human hairstyles. The Cosserat Rod model requires only the setting of a few parameters to generate realistic results.

Another advantage of the Cosserat Rod over the alternatives is that the method represents a smooth deformable model. This deformable model can be sampled directly and does not have to be tessellated into infinitesimal line segments first. Nodal hair modelling techniques require tessellation, for example to a Bezier or a B-spline, to smoothen out the hard corners fount at the joints where two elements of hair connect.

In the paper of (Bertails, et al., 2005b) the energy minimization of the Cosserat Rods is performed by the Davidon–Fletcher–Powell method described in (Fletcher, et al., 1963). The Davidon-Fletcher-Powell optimization algorithm belongs to the family of first order optimization algorithms known as gradient descent or steepest-descent, which means that the algorithm finds a local minimum of a given function by iteratively traversing the parameter space with the path that has the steepest gradient. This process is continued until the gradient converges to zero.

While the Davidon-Fletcher-Powel algorithm is performing well in (Bertails, et al., 2005b), I have taken another approach in this project. I have chosen to use the Rosenbrock method instead. The Rosenbrock method, explained in (Rosenbrock, 1960), is a zero order search algorithm, which means that it does not require any derivatives or gradients from the basic function that needs to be optimized. While only simple evaluations of the basic function are required, it can approximate a gradient search; thus combining advantages of zero order and first order strategies. My opinion is that this algorithm is particularly well suited for the Cosserat Rod, because the derivatives of the energy function of the Cosserat Rod are more complex to calculate than the original energy functions; while more evaluations are needed with the Rosenbrock method, it will lead, in the long end, to a more stable method due to how the derivative of a function is approximated.

Although the calculation of the shape of a single hair strand can be performed within a fraction of a second, the time begins to add up when one wants to calculate an entire hair coupe of a person with around 100 000 to 150 000 hair strands.

One solution, proposed by (Daldegan, et al., 1993), considering that the hair has an overall coherence in its appearance during the movement, one can simply animate only a few guide hairs, then expand this movement to the rest of the hair by interpolation. The disadvantage is that it only applies to very smooth hair, and, also, shows no discontinuity in the movement. In addition, this method poses the problem of collisions between the hair and body, since only a few hairs are able to detect interpenetration.

Another solution is to consider that hair has not an overall continuity, but rather has local internal cohesion. The hair is then modelled as a set of geometric primitives called wisps, which are animated separately. Obviously, this model also reduces the number of individual elements to animate, while keeping some complexity to the hair. Several studies have been based on this approach, some simulating a small number of coarse wisps, others using surfaces for wisps, such as (Koh, et al., 2001) and (Koh, et al., 2000), or volumes for wisps, such as (Noble, et al., 2004), these wisps can be deformable or rigid.

The model that this project uses is a combination of the two solutions described above. To be able to handle both smooth and clumpy hairstyles, I avoid choosing between a continuum and a wisp-based representation for hair. Many real hairstyles display an intermediate behaviour with hair strands being more evenly spaced near the scalp than near the tips. My solution, based on (Bertails, et al., 2006) uses a semi-interpolating scheme to generate non-simulated hair strands from the guide strands; the framework ranges from full interpolation to generate the extra hair strands near the scalp to full extrapolation within a hair wisp near the tips. The separation strongly depends on the level of curliness: straight hair requires relatively more interpolation – compared to extrapolation – than curly and clumpy hair.

Finally, illumination is taken care of by a local illumination model introduced by (Kajiya, et al., 1989), and shadows are added by implementing the deep opacity maps developed by (Yuksel, et al., 2008).

(Kajiya, et al., 1989) has two major advantages over other illumination models. The first advantage is that it is the single simplest shading algorithm to understand and to implement. And second, due to its simplicity, it requires only few calculations and is thus also a very fast algorithm for hair illumination. (Kajiya, et al., 1989) motivates that although the algorithm is not entirely physical correct, it can produce images of adequate quality.

The reason for choosing to implement the shadow technique of (Yuksel, et al., 2008), is that (Yuksel, et al., 2008) have proven, through a comparison of several shadow techniques, that their technique is superior over others in both the performance of the algorithm and the quality of the shadows generated.

Now that I have give a global overview of my hair simulation & rendering framework, the following sections will give a comprehensive description of the techniques implemented by the framework. The techniques that will be explained are: the Cosserat Rods, Hair Collision, Energy Calculation, Energy Minimization, Hair Interpolation and Hair Illumination.

4.1.1. Cosserat Rods

One of the newest methods – based on physics – for simulating the shape of a hair strand, is developed by (Bertails, et al., 2005b). This paper uses the Kirchhoff equations for static Cosserat Rods to calculate the shape of a hair strand. The method is based upon a mechanically accurate model for static elastic rods, which accounts for the natural curliness of hair.

The model of the Cosserat rod is a mathematical representation of a rod material. In this context, a rod material can be explained as a slender object whose dimensions in the two directions of the plane of the cross section are negligible compared to the longitudinal dimension. Hair is thus a good example of a rod material.

The Kirchhoff equations describe the deformation of these elastic inextensible rods. These equations, although made in the context of Hooke's linear law of elasticity, are nonlinear and can describe the changes in form from the natural configuration of a rod. The Kirchhoff equations take into account all modes of deformation observed in real hair (bending, torsion and longitudinal inextensibility). In addition, all parameters involved in these equations are related to simple properties such as the hair natural curvatures, hair ellipticity and hair bending stiffness.

Although the model is complex, it yields various typical hair configurations that can also be observed in the real world, such as ringlets. The method is able to account for the structural properties of the different hair types existing in the world – Asian, African and Caucasian hair – by setting very few input parameters. This method is also able to simulate the non-linear instabilities in real hair, which are mentioned earlier in section ***Mechanical Properties***, located in chapter ***Natural Hair Properties***.

The Super-Helix model – developed in (Bertails, et al., 2005b), (Bertails, et al., 2006) and (Bertails, 2009) – is a discrete implementation of the Cosserat Rod, and is the model that I used in the hair simulation & rendering framework to represent the shape of the individual hair strands at rest. The Super-Helix model will be explained in the following sub-sections.

4.1.1.1. Super-Helices Introduction

The definition of a Super-Helix starts with the inextensible Cosserat Rod of length L. The value $s \in [0, L]$ forms the curvilinear abscissa along this rod; a curvilinear abscissa is the distance on the curve relative to its starting point. The centreline, represented by $\boldsymbol{r}(s)$, is the curve passing through the centre of mass of every cross section of the rod. This curve in space can describe the shape of the rod, but cannot recognize the amount of twist that the rod has around its centreline. In order to keep track of this twist, the Cosserat model introduces a material frame, represented by $\boldsymbol{n}_i(s)$ – where $i = 0,1,2$ – at every point of the centreline, that 'flows' along with the surrounding material upon deformation of the centreline. By convention, $\boldsymbol{n}_0$ is the tangent $\boldsymbol{t}$ to the centreline:

$$\boldsymbol{r}'(s) = \boldsymbol{n}_0(s) = \boldsymbol{t}(s)$$

While $\boldsymbol{n}_\alpha$ – where $\alpha = 1,2$ – are the other two vectors that span the plane of the cross section.

Figure 15 shows an illustration of the Cosserat Rod. Every point of a rod contains four vectors: first $\boldsymbol{r}(s)$, which represents a position in a three-dimensional space, a vector $\boldsymbol{n}_0(s)$, which represents the tangent, and two vectors $\boldsymbol{n}_1(s)$ and $\boldsymbol{n}_2(s)$ which represent the plane of the cross section.

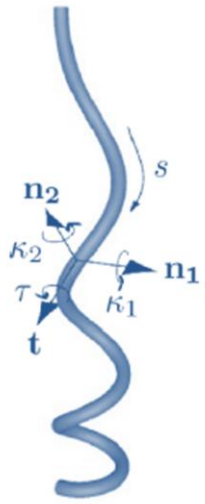

Figure 15. Cosserat Rod

The Kirchhoff model for elastic rods starts from this mathematical description of a Cosserat curve and incorporates the physical requirement of inextensibility and unshearability, which can also be found in natural hair. In this particular case, the material frame $\boldsymbol{n}_i(s)$ – where $i = 0,1,2$ – is orthonormal for all s, and there exists a vector $\boldsymbol{\Omega}(s)$, which is called the Darboux vector, that defines the change of the material frame, such that:

$$\boldsymbol{n}'_i(s) = \boldsymbol{\Omega}(s) \times \boldsymbol{n}_i(s)$$

Inextensibility of the hair strand is established by the fact that the material frame is represented by normalized unit vectors, and unshearability is enforced because the vectors of the material frame are required to be orthogonal to each other.

Appropriate boundary conditions are specified: one end of the hair strand, $s = 0$, is clamped into the head while the other end, $s = L$, is free. The positions of the clamped end, together with the orientation of the initial frame, are imposed by head motion:

$$\boldsymbol{r}(0) = \boldsymbol{r}_c$$

$$\boldsymbol{n}_i(0) = \boldsymbol{n}_{i,c}$$

where subscript 'c' refers to the clamped end of the rod, $s = 0$.

The rod's material curvatures $\kappa_\alpha(s)$ – where $\alpha = 1,2$ – with respect to the two directions of the cross section and the twist $\tau(s)$ are defined as the coordinates of the Darboux vector $\boldsymbol{\Omega}(s)$ in the local material frame. The vector can be calculated with the following equation:

$$\boldsymbol{\Omega}(s) = \tau(s)\,\boldsymbol{n}_0(s) + \kappa_1(s)\,\boldsymbol{n}_1(s) + \kappa_2(s)\,\boldsymbol{n}_2(s)$$

By introducing a redundant notation for the twist, $\kappa_0 = \tau$, one can refer to these parameters collectively as $\kappa_i(s)$ – where $i = 0,1,2$ – and the above formula can be reformulated as:

$$\boldsymbol{\Omega}(s) = \sum_{i=0,1,2} \left(\kappa_i(s)\,\boldsymbol{n}_i(s)\right)$$

The degrees of freedom of a Kirchhoff rod are its material curvatures and twist $\kappa_i(s)$ – where $i = 0,1,2$ – along the curvilinear abscissa. Each vector of the material frame is affected by two material curvatures. The relationships between material frame and material curvature can be calculated from the equations listed above. The relations are:

$$\boldsymbol{n}'_0(s) = \boldsymbol{\Omega}(s) \times \boldsymbol{n}_0(s) = \kappa_2(s)\,\boldsymbol{n}_1(s) - \kappa_1(s)\,\boldsymbol{n}_2(s)$$

$$\boldsymbol{n}'_1(s) = \boldsymbol{\Omega}(s) \times \boldsymbol{n}_1(s) = \kappa_0(s)\,\boldsymbol{n}_2(s) - \kappa_2(s)\,\boldsymbol{n}_0(s)$$

$$\boldsymbol{n}'_2(s) = \boldsymbol{\Omega}(s) \times \boldsymbol{n}_2(s) = \kappa_1(s)\,\boldsymbol{n}_0(s) - \kappa_0(s)\,\boldsymbol{n}_1(s)$$

From the equations listed above, it becomes clear that every material frame vector $\boldsymbol{n}_i$ is perturbed in perpendicular directions $\boldsymbol{n}_j$ and $\boldsymbol{n}_k$, by an amount specified by κ_k and κ_j respectively – where $i \neq j \wedge j \neq k \wedge k \neq i$. This can also be seen intuitively in Figure 15; for example when imagining a twist τ around the axis of the rod, the vectors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ will spin with it, while the vector $\boldsymbol{n}_0$ or $\boldsymbol{t}$ will stay unaffected.

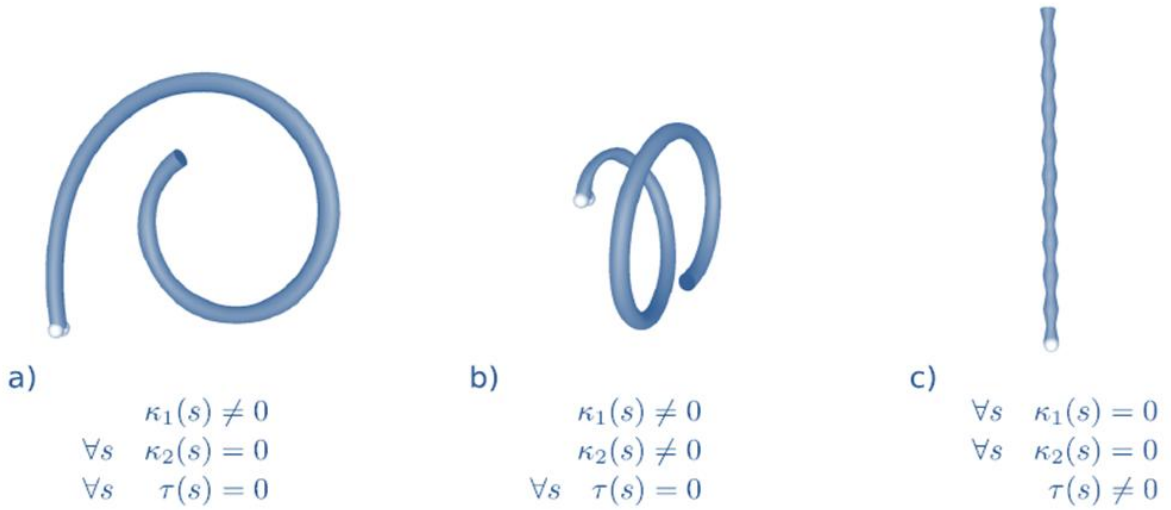

Figure 16. Bending and Twisting of Kirchhoff Rod

Figure 16 illustrates the effects that different material curvatures and twists can have on the shape of the Kirchhoff rod. In situation **A**) the twist and one of the two bends remains zero, which results in a spiral resting in a plane. In situation **B**) only the twist remains zero, which results in a spiral that leaves the plane. And in situation **C**) the two bends remain zero, resulting in twisted straight rod. When all curvatures and twist would remain zero, the result would be an untwisted straight rod. Any arbitrary shape can be modelled by varying the curvatures and twist along the curvilinear abscissa.

The Kirchhoff model is a continuous model and requires that every point, along the curvilinear abscissa, of the rod has specified two curvatures and one twist. A numerical model of the Kirchhoff equations would be impractical for Computer Graphics to implement, because the numerical model would require an almost infinite number of infinite small elements to remain stable; the slightest change in value of the material curvatures and twist will have a big impact on the overall shape of the rod. The infinite amount of elements would result in a large amount of time being spent on calculations. (Bertails, et al., 2005b), (Bertails, et al., 2006) and (Bertails, 2009) introduced another approach of discretion that uses symbolic calculations which results in a faster and more stable implementation of the Kirchhoff equations. This form of discretization transformed the Kirchhoff model, derived by Kirchhoff in 1859, into the Super-Helix model.

A Super-Helix is composed of N helical elements E_Q indexed by Q $(1 \leq Q \leq N)$, each element being parameterized by a constant twist $\kappa_0 = \tau$ and two constant curvatures κ_1 and κ_2. The full Super-Helix is thus parameterized by the $3N$ generalized coordinates $\kappa_{i,Q}(t)$, where $i = 0,1,2$ specifies the mode of deformation for each element. The vector of size $3N$ collecting all the generalized coordinates is denoted $\boldsymbol{\kappa}$, while the three functions $\kappa_i(s,t)$, $i = 0,1,2$ stand for the piecewise constant twist and curvature functions along the rod:

$$\kappa_i(s,t) = \sum_{1 \leq Q \leq N} \left(\chi_Q(s) \kappa_{i,Q}(t) \right) \quad for\ i = 0,1,2$$

where $\chi_Q(s)$ equals 1 for $s \in E_Q$ and 0 elsewhere.

The generalized coordinates $\boldsymbol{\kappa}$ can be used to reconstruct the rod. The equations listed in this section can be combined to form a differential equation with respect to s. By integration of this equation, one obtains the material frame $\boldsymbol{n}_i(s)$. $\boldsymbol{n}_0(s)$ can then be integrated to obtain the centreline $\boldsymbol{r}(s)$ of the Cosserat Rod. Both, material frame $\boldsymbol{n}_i(s)$ and centreline $\boldsymbol{r}(s)$ are a function of s and $\boldsymbol{\kappa}$. The reconstruction of $\boldsymbol{n}_i(s)$ and $\boldsymbol{r}(s)$ will be described in the next section.

(Bertails, et al., 2005b), (Bertails, 2006), (Bertails, et al., 2006), (Bertails, 2009)

4.1.1.2. Super-Helix Reconstruction

In the previous section mentioned integration can be carried out analytically; the integration with respect to s has a symbolic solution over every element E_Q of the Super-Helix. By patching the solutions together, the model deforms as a helix over every element and is C_1-smooth[3] over the rod; between adjacent elements, both the centreline and the material frames are continuous.

[3] The class C_n-smooth consists of all the differentiable functions whose n^{th} derivative is continuous; C_1-smooth functions are called continuously differentiable.

A Super-Helix is divided into N helical elements E_Q indexed by Q $(1 \leq Q \leq N)$, where the element $E_Q = \{s, s_{Q-1} \leq s \leq s_Q\}$. In this definition, the value s_Q represents the curvilinear abscissa at the end of element E_Q. This means that $s_0 = 0$ and $s_N = L$.

Let ℓ_Q be the length of the element E_Q:

$$\ell_Q = s_Q - s_{Q-1}$$

For simplicity in the remainder of the section, the value $u \in [0, \ell_Q]$ denotes the curvilinear abscissa of an element E_Q. This value can be calculated with the following equation.

$$u = s - s_{Q-1}$$

To preserve C_1-smoothness over the rod, both the centreline and the material frames must be continuous between adjacent elements. Thus for $\forall Q \geq 2$ the following relationship holds:

$$\boldsymbol{n}_{i,Q}(0) = \boldsymbol{n}_{i,Q-1}(\ell_{Q-1})$$

$$\boldsymbol{r}_Q(0) = \boldsymbol{r}_{Q-1}(\ell_{Q-1})$$

Figure 17 summarizes the main notations used for an element E_Q.

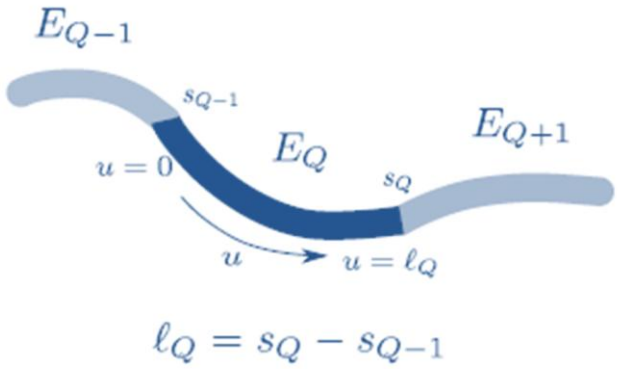

Figure 17. Super-Helix Element

When considering that each element of the Super-Helix is being parameterized by a constant twist $\kappa_0 = \tau$ and two constant curvatures κ_1 and κ_2, it can be shown that the Darboux vector stays constant along the element. This would mean that the change of the Darboux vector along the curvilinear abscissa remains zero.

Starting from the equation that calculates the Darboux vector:

$$\boldsymbol{\Omega}(u) = \sum_{i=0,1,2} \left(\kappa_i \boldsymbol{n}_i(u)\right)$$

I take the derivative of the Darboux vector with respect to u:

$$\boldsymbol{\Omega}'(u) = \sum_{i=0,1,2} \left(\kappa_i \boldsymbol{n}_i'(u)\right)$$

Because $\boldsymbol{n}'_i(s) = \boldsymbol{\Omega}(s) \times \boldsymbol{n}_i(s)$, the equation becomes:

$$\boldsymbol{\Omega}'(u) = \sum_{i=0,1,2} \Big(\kappa_i\big(\boldsymbol{\Omega}(u) \times \boldsymbol{n}_i(u)\big)\Big)$$

Because a cross-product is distributive over addition, the equation becomes

$$\boldsymbol{\Omega}'(u) = \boldsymbol{\Omega}(u) \times \sum_{i=0,1,2} \big(\kappa_i \boldsymbol{n}_i(u)\big)$$

And because $\boldsymbol{\Omega}(s) = \sum_{i=0,1,2}\big(\kappa_i(s)\boldsymbol{n}_i(s)\big)$, the equation becomes:

$$\boldsymbol{\Omega}'(u) = \boldsymbol{\Omega}(u) \times \boldsymbol{\Omega}(u)$$

Now it becomes clear that the first order derivative of the Darboux vector remains zero along the entire element, because a cross-product between similar vectors will result in $\mathbf{0}$ by definition. Thus the derivative of the Darboux vector becomes:

$$\boldsymbol{\Omega}'(u) = \mathbf{0} \quad for\ \forall u \in \big[0, \ell_Q\big]$$

As the first order derivative of the Darboux vector is $\mathbf{0}$, The Darboux vector will be constant along each element. For a given element E_Q, one can therefore use Ω the norm of the vector $\boldsymbol{\Omega}$ and $\boldsymbol{\omega} = \boldsymbol{\Omega}/\Omega$ the unit vector aligned with $\boldsymbol{\Omega}$. Finally, we write $\boldsymbol{a}^{\parallel} = (\boldsymbol{a} \cdot \boldsymbol{\omega})\boldsymbol{\omega}$ and $\boldsymbol{a}^{\perp} = \boldsymbol{a} - \boldsymbol{a}^{\parallel}$ as the projection of an arbitrary vector a parallel to and perpendicular to the axis spanned by $\boldsymbol{\omega}$, respectively.

Since $\boldsymbol{\Omega}$ is constant, and can be seen as an angular velocity vector, integration of equation $\boldsymbol{n}'_i(u)$ over an element becomes straightforward. The material frame 'rotates' around $\boldsymbol{\omega}$ with a constant rate of rotation Ω per unit of curvilinear length. Therefore, the material frame at coordinate $u \in E_Q$ is obtained from the material frame $\boldsymbol{n}_{i,Q}(0)$ given on the left-hand side of the interval $u = 0$, by a rotation with angle $\Omega * u$ and axis parallel to $\boldsymbol{\omega}$:

$$\boldsymbol{n}_{i,Q}(u) = \boldsymbol{n}^{\parallel}_{i,Q}(0) \ + \boldsymbol{n}^{\perp}_{i,Q}(0) \ \cdot cos(\Omega\, u) + \boldsymbol{\omega} \times \boldsymbol{n}^{\perp}_{i,Q}(0) \ \cdot sin(\Omega\, u)$$

Because $\boldsymbol{r}'(u) = \boldsymbol{n}_0(u)$, the centreline $\boldsymbol{r}_Q(s)$ can be found by the spatial integration of the tangent, $\boldsymbol{n}_{0,Q}(u)$:

$$\boldsymbol{r}_Q(u) = \boldsymbol{r}_Q(0) + \boldsymbol{n}^{\parallel}_{0,Q}(0)\, u + \boldsymbol{n}^{\perp}_{0,Q}(0) \cdot \frac{sin(\Omega\, u)}{\Omega} + \boldsymbol{\omega} \times \boldsymbol{n}^{\perp}_{0,Q}(0) \cdot \frac{1 - cos(\Omega\, u)}{\Omega}$$

Where $\boldsymbol{r}_Q(0)$ is the prescribed position of the centreline on the left-hand side of the interval.

Above equations provide the explicit reconstruction of an element. Its centreline is a helix with axis parallel to $\boldsymbol{\omega}$. Two degenerate cases are possible and must be considered separately: the curve is an arc of circle when $\tau = 0$ and $\kappa_1 \neq 0$ or $\kappa_2 \neq 0$; it is a straight line when $\kappa_1 = \kappa_2 = 0$, which can be twisted ($\tau \neq 0$) or untwisted ($\tau = 0$, implying $\boldsymbol{\Omega} = \mathbf{0}$).

Figure 18 shows an illustration of the reconstruction of an element. In this illustration vectors v_0 and v_1 represents $\boldsymbol{n}^{\parallel}_{i,Q}(0)$ and $\boldsymbol{n}^{\perp}_{i,Q}(0)$, which are the projection of $\boldsymbol{n}_{i,Q}(0)$ parallel to and perpendicular

to the axis spanned by $\boldsymbol{\omega}$, respectively. The vector v_2 represent $\boldsymbol{\omega} \times \boldsymbol{n}_{i,Q}^{\perp}(0)$, which is the vector perpendicular to v_0 and v_1.

For the calculation of the three vectors $\boldsymbol{n}_{i,Q}(u)$, the component of $\boldsymbol{n}_{i,Q}(0)$ in the direction of v_0 remains constant over u, while the components of $\boldsymbol{n}_{i,Q}(0)$ in the direction of v_1 and v_2 spins around the axis of the Darboux vector $\boldsymbol{\Omega}$ with a velocity equal to the norm of the Darboux vector, which is Ω. The rotation can be calculated through the use of trigonometric functions.

For the calculation of $\boldsymbol{r}_Q(u)$, the component of $\boldsymbol{n}_{i,Q}(0)$ in the direction of v_0 is multiplied with u, which results in a position that moves along in the direction of the Darboux vector $\boldsymbol{\Omega}$, while the two components of $\boldsymbol{n}_{i,Q}(0)$ in the directions v_1 and v_2 orbits around the axis of the Darboux vector $\boldsymbol{\Omega}$ with a velocity equal to the norm of the Darboux vector, which is Ω. This rotation is also achieved through the use of trigonometric functions. The starting point of the helix element is determined by $\boldsymbol{r}_Q(0)$, and the radius of the helix is determined by dividing the trigonometric functions with the norm of the Darboux vector Ω.

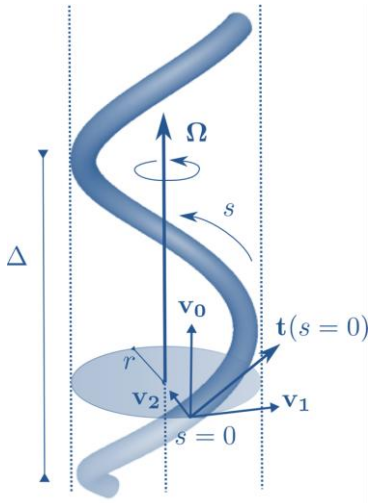

Figure 18. Super Helix Reconstruction

At this point the constant Darboux vector $\boldsymbol{\Omega}$ of an element can be calculated, when the three $\boldsymbol{n}_{i,Q}(0)$ of that element are known, with:

$$\boldsymbol{\Omega} = \sum_{i=0,1,2} \left(\kappa_{i,Q} \boldsymbol{n}_{i,Q}(0) \right)$$

This Darboux vector and the three $\boldsymbol{n}_{i,Q}(0)$ can then be plugged into $\boldsymbol{n}_{i,Q}(u)$ to calculate the Darboux frame of the element along the curvilinear abscissa, and when $\boldsymbol{r}_Q(0)$ is known, it becomes possible with equation $\boldsymbol{r}_Q(u)$ to calculate the centreline of the element along the curvilinear abscissa.

As mentioned earlier, because one end of the hair strand, $s = 0$, is clamped into the head, the position and orientation of the initial frame of the clamped end are imposed by head motion:

$$\boldsymbol{r}_1(0) = \boldsymbol{r}_c$$

$$\boldsymbol{n}_{i,1}(0) = \boldsymbol{n}_{i,c}$$

$\boldsymbol{n}_{i,1}(0)$ and $\boldsymbol{r}_1(0)$ of element E_1 are thus specified and can be used to reconstruct the centreline and material frames of the first element. With the smoothness conditions:

$$\boldsymbol{n}_{i,Q}(0) = \boldsymbol{n}_{i,Q-1}(\ell_{Q-1})$$

$$\boldsymbol{r}_Q(0) = \boldsymbol{r}_{Q-1}(\ell_{Q-1})$$

$\boldsymbol{n}_{i,2}(0)$ and $\boldsymbol{r}_2(0)$ of element E_2 are thus specified and can be used to reconstruct the next element. This process can be repeated till one reaches the last element of the Super Helix. By recursively patching the helix elements together one can reconstruct the entire Super-Helix, starting from the clamped end $s = 0$ and continuing until the free end $s = L$ is reached.

A Super-Helix is a realistic, stable and efficient hair simulation model. All integrations are performed symbolically offline, leading to a quick and accurate evaluation of the coefficients in the equation. The inextensibility constraints, enforced by a unity Darboux frame, are incorporated into the reconstruction process. This means that all generalized coordinates in vector $\boldsymbol{\kappa}$ are free of any constraints and the stiff constraints of inextensibility have been removed from the equations.

The method allows for a well-controlled space discretization, leading to stable simulations even for a small number of elements N. When $N \to \infty$ the discretion becomes so small that the original Kirchhoff equations are recovered, making the simulation very accurate. By tuning the number of elements, one can freely choose the best compromise between accuracy and efficiency, depending on the complexity of the hairstyle and the allowed computational time. Typically a value of $N = 5$ elements is sufficient to model believable human hair. As every element has three degrees of freedom, the total hair strand has $3N = 15$ degrees of freedom.

(Bertails, 2006), (Bertails, et al., 2006)

4.1.1.3. Super-Helix Potential Energy

As mentioned in earlier chapters, the formation of a hair shaft can be seen as an extrusion from the hair follicle, and its shape remains relatively constant along the entire strand. This means that the hair strand has an almost uniform cross section, a natural twist and natural curvatures all along the shaft. These twist and curvatures are responsible for the shape of the hair and will determine whether the hair becomes straight, curly, or fuzzy.

This intrinsic curliness of the hair can be expressed by natural torsion and curvatures τ^0, κ_1^0 and κ_2^0, which describes the state of the rod when the hair is not subject to external forces. Because the natural curvatures and torsion are roughly constant along the entire hair strand, the configuration of a hair strand in the absence of external forces becomes a plain helix.

However, when the hair is subject to a gravity field, the shape of the hair strand at rest will be an equilibrium between the downward pull of the gravity field and the elastic properties of the hair trying to recover the hair strand to a naturally helical shape. The stable equilibrium shape of the hair strand can be found by searching for the configuration that minimizes the potential energy stored within the hair. The potential energy stored within the hair strand can be formulated as:

$$\mathcal{E}_{hair} = \mathcal{E}_g + \mathcal{E}_e$$

where $\mathcal{E}_e$ is the internal elastic energy of the rod and $\mathcal{E}_g$ the energy of the rod accounting for gravity.

Figure 19 contains three screenshots taken from within the hair framework. These three illustrations show – from left to right – energy minimization with:

a) only the internal elastic energy $\mathcal{E}_e$, resulting in a hair strand deforming like a plain helix with natural torsion and curvatures τ^0, κ_1^0 and κ_2^0.

b) only the potential gravitational energy $\mathcal{E}_g$, resulting in a hair strand hanging straight down to the ground.

c) both the internal elastic energy $\mathcal{E}_e$ and the potential gravitational energy $\mathcal{E}_g$, resulting in an equilibrium between these two energies. The hair strand forms a ringlet.

The red and yellow colours in the illustrations represent the different elements of the Super-Helix which is used to model the shape of the hair strand.

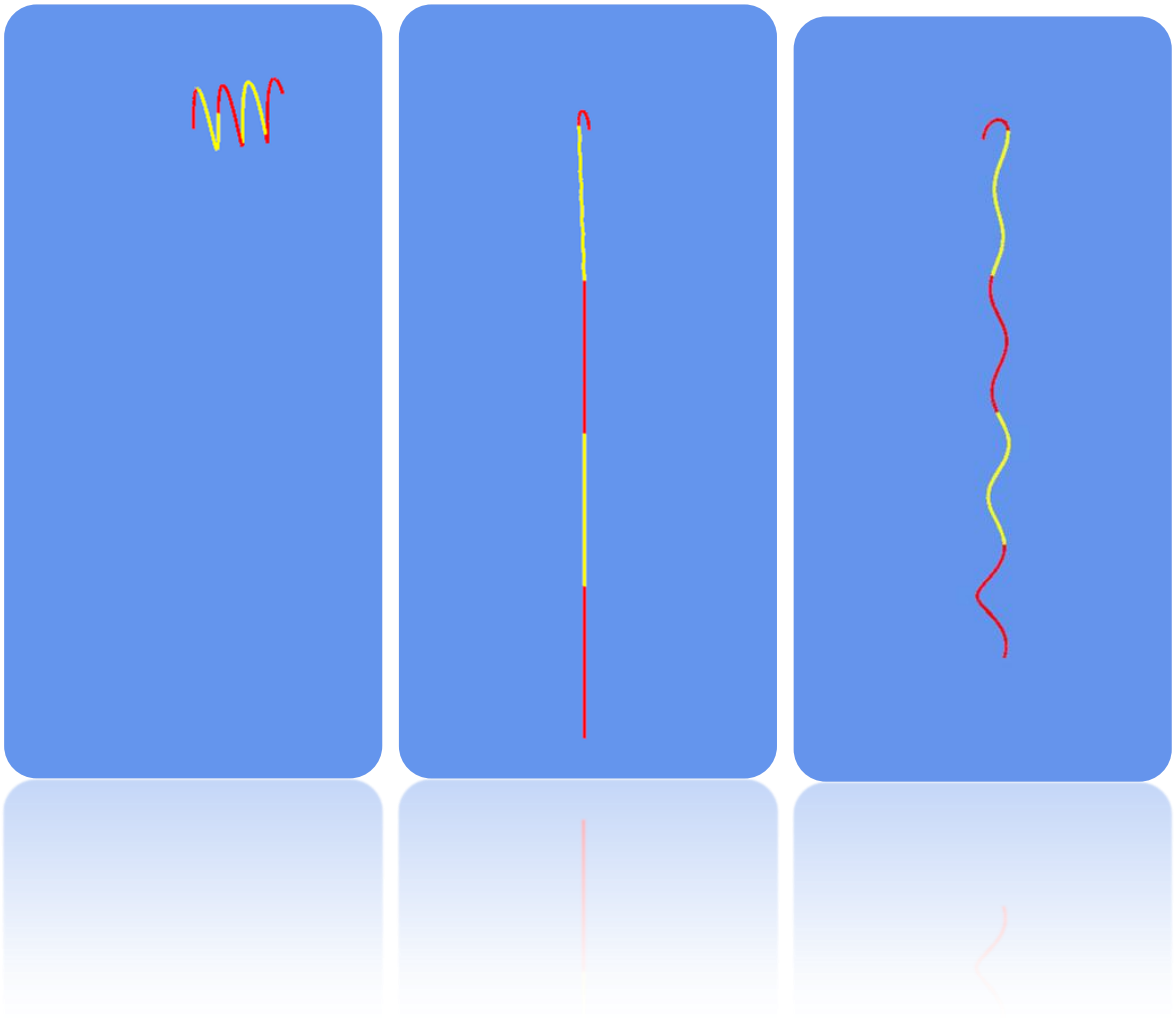

Figure 19. Energy Minimization

The equations used for calculating the internal elastic energy $\mathcal{E}_e$ and the potential gravitational energy $\mathcal{E}_g$ will be explained in the following sub sections.

4.1.1.3.1. Elastic Energy

Assuming that the hair strand is a rod with an elliptic cross section and obeying Hooke's law for elasticity, (Bertails, et al., 2005a) proposes that $\mathcal{E}_e$, the internal elastic energy, can be written as:

$$\mathcal{E}_e = \int_0^L \left[\frac{EI_1}{2}(\kappa_1(s) - \kappa_1^0)^2 + \frac{EI_2}{2}(\kappa_2(s) - \kappa_2^0)^2 + \frac{\mu J}{2}(\tau(s) - \tau^0)^2 \right] ds$$

where E is the Young's modulus, μ is the shearing modulus, I_1 and I_2 are the momentum of inertia of the rod's cross section with $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ respectively, and J is the axial momentum of inertia.

The Young's modulus and shearing modulus are both modulus of elasticity. The Young's modulus is a measure of stiffness of an elastic material; it describes the material's response to linear strain, while the shearing modulus is a measure of shear strain; it describes the material's response to shearing strains. The shearing modulus μ can be calculated from the Young's modulus E with:

$$\mu = \frac{E}{2(1+\sigma)}$$

where σ is the Poisson's ratio of the material. When a piece of material is stretched in one direction it tends to get thinner in the other two directions. The Poisson's ratio is the ratio of transverse contraction strain to the longitudinal extension strain in the direction of stretching force.

The momentum of Inertia is a measure of an object's resistance to changes in its rotation rate. It is dependent on the shape of the material. Because the cross section of a hair strand has an elliptic shape, the equation for the moment of inertia of an ellipse is taken:

$$I_1 = \frac{\pi * ab^3}{4}$$

$$I_2 = \frac{\pi * ba^3}{4}$$

$$J = \frac{\pi * ab}{4}(a^2 + b^2)$$

The momentum of Inertia depends on the principal radii of the ellipse, in the equations above a represents the major axis of the ellipse along $\boldsymbol{n}_1$, and b represents the minor axis of the ellipse along $\boldsymbol{n}_2$. The ellipticity e of the eccentricity is calculated with:

$$e = \sqrt{\frac{a^2 - b^2}{a^2}}$$

where a and b follow from the average radius r of the hair strand:

$$r = \frac{a+b}{2}$$

The eccentricity of an ellipse ranges from 0 to 1, where a perfect round circle has an eccentricity of 0, and an increasingly flattened ellipse has an eccentricity going to 1.

Table 1 contains a list of physical measurements done by (Bertails, et al., 2006). This table contains natural quantities for average radius r, eccentricity e, Poisson's ratio $\boldsymbol{\sigma}$, and Young's modulus E. The table contains the natural quantities of the three broad categories of hair in the world; Asian, Caucasian and African. These values are used as input for the equations above, needed to calculate the internal elastic energy $\mathcal{E}_e$ of the hair strand.

Table 1. Physical Quantities of Natural Hair 1

	Asian	Caucasian	Caucasian	African
(r) Radius	$50\ \mu m$	$35\ \mu m$	$50\ \mu m$	$50\ \mu m$
(e) Eccentricity	0	0.17	0.17	0,31
(σ) Poisson's ratio	0.48	0.48	0.48	0.48
(E) Young's modulus	$1\ GPa$	$2\ GPa$	$1.5\ GPa$	$0.5\ GPa$

The natural intrinsic curliness of the hair, τ^0, κ_1^0 and κ_2^0, are roughly constant along the entire hair strand, and can be specified by the following equations:

$$\tau^0 = \frac{\Delta_h}{2\pi\, r_h^2}$$

$$\kappa_1^0 = \frac{1}{r_h}$$

$$\kappa_2^0 = 0$$

In these equations, r_h and Δ_h represent the radius and step size of the helix. These quantities are visually represented in Figure 20. r_h and Δ_h specifies the helical shape to which the hair returns when elasticity is the only force acting on the hair strand.

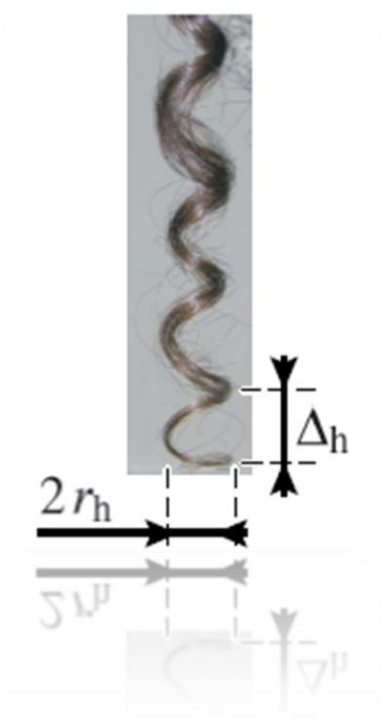

Figure 20. Helix Radius and Step Size

Table 2. shows some typical values for radius r_h and step size Δ_h used to represent smooth, wavy, curly or fuzzy hair.

Table 2. Physical Quantities of Natural Hair 2

	Smooth	Wavy	Curly	Fuzzy
(r_h) Helix Radius	$0.0\ cm$	$1.0\ cm$	$0.6\ cm$	$0.1\ cm$
(Δ_h) Helix Step	$0.0\ cm$	$0.5\ cm$	$0.5\ cm$	$1.0\ cm$

4.1.1.3.2. Gravitational Energy

(Bertails, et al., 2005a) proposes that $\mathcal{E}_g$, the potential gravitational energy, can be written as:

$$\mathcal{E}_g = \rho S g \int_0^L z(s)\, ds$$

where ρ is the volumic mass of the rod, S the area of its cross section, g the gravity field value and $z(s)$ the vertical coordinate of element ds at curvilinear abscissa s.

The area of the cross section is dependent on the shape of the cross section. Because the cross section of a hair strand has an elliptic shape, the equation for the area of an ellipse is taken:

$$S = \pi * ab$$

Table 3 contains a list of physical quantities. This table contains the natural quantities for hair density $\boldsymbol{\rho}$, and earth's gravity field value $\boldsymbol{g}$. These values are used as input for the equations above, needed to calculate the potential gravitational energy $\mathcal{E}_g$ of the hair strand.

Table 3. Physical Quantities of Natural Hair 3

	Constants
(ρ) Hair Density	$1.3\ g/cm^3$
(g) Gravity Field	$9{,}81\ N/kg$

Although the internal elastic energy $\mathcal{E}_e$ and the potential gravitational energy $\mathcal{E}_g$ of the hair strand are known, they cannot yet be directly used in the implementation of the hair simulation & rendering framework. The integrals of both equations need to be solved.

For the elastic energy, the internal elastic energy $\mathcal{E}_e$ can be rewritten as the sum of elastic energy stored within every element E_q of the hair strand:

$$\mathcal{E}_e = \sum_{Q=1}^{N} \int_0^{\ell_Q} \left[\frac{EI_1}{2}(\kappa_1 - \kappa_1^0)^2 + \frac{EI_2}{2}(\kappa_2 - \kappa_2^0)^2 + \frac{\mu J}{2}(\tau - \tau^0)^2 \right] ds$$

Because the torsion τ and two curvatures κ_1 and κ_2 are constant along the curvilinear abscissa from 0 to ℓ_Q of a single element $\mathcal{E}_{e,Q}$, integration becomes straight forward and the result of the integration is:

$$\mathcal{E}_e = \sum_{Q=1}^{N} \left[\left(\frac{EI_1}{2}(\kappa_1 - \kappa_1^0)^2 + \frac{EI_2}{2}(\kappa_2 - \kappa_2^0)^2 + \frac{\mu J}{2}(\tau - \tau^0)^2 \right) u \right]_0^{\ell_Q}$$

which result in an implementable elastic energy equation:

$$\mathcal{E}_e = \sum_{Q=1}^{N} \left(\frac{EI_1}{2}(\kappa_1 - \kappa_1^0)^2 + \frac{EI_2}{2}(\kappa_2 - \kappa_2^0)^2 + \frac{\mu J}{2}(\tau - \tau^0)^2 \right) \ell_Q$$

This equation calculates the difference between the natural and actual torsion and curvature of the super-helix. The quadratic difference is multiplied with a stiffness penalty, which depends on the stiffness of the hair material and the shape of the hair cross section. The more a hair deviates from its natural torsion and curvature, the larger the internal elastic energy of the hair strand becomes. When the hair strand has a natural torsion and curvature, the energy will become zero.

The potential gravitational energy is more difficult to be rewritten to an implementable version. Knowing that $z(s)$ is the vertical coordinate of element ds at curvilinear abscissa s, $z(s)$ can be replaced with $\boldsymbol{r}(s) \cdot \boldsymbol{e}_{vert}$:

$$\mathcal{E}_g = \rho S g \int_0^L [\boldsymbol{r}(s) \cdot \boldsymbol{e}_{vert}]\, ds$$

where $\boldsymbol{e}_{vert}$ is a constant vector pointing along the vertical axis of the world. As $\boldsymbol{e}_{vert}$ is constant, it can be pulled out of the integral:

$$\mathcal{E}_g = \rho S g \int_0^L [\boldsymbol{r}(s)]\, ds \cdot \boldsymbol{e}_{vert}$$

and because $\boldsymbol{r}(s)$ is the integral of $\boldsymbol{n}_0(s)$**:**

$$\mathcal{E}_g = \rho S g \int_0^L \left[\int_0^s [\boldsymbol{n}_0(s')]\, ds' \right] ds \cdot \boldsymbol{e}_{vert}$$

The function $f(s, s') = \boldsymbol{n}_0(s')$ is continues and bounded by the triangular domain:

$$\Delta = \{(s', s) \mid 0 \leq s' \leq s \leq L\}$$

By Fubini's Theorem, $f(s, s')$ can therefore be integrate over domain Δ by choosing any order of integration. By exchanging the order of integrals in the above integral, the equation becomes:

$$\mathcal{E}_g = \rho S g \int_0^L \left[\boldsymbol{n}_0(s') \int_{s'}^L ds \right] ds' \cdot \boldsymbol{e}_{vert}$$

because $\int_{s'}^L ds = [s]_{s'}^L = (L - s')$:

$$\mathcal{E}_g = \rho S g \int_0^L [\boldsymbol{n}_0(s')\ (L - s')]\, ds' \cdot \boldsymbol{e}_{vert}$$

Remembering that $u = s - s_{Q-1}$, the potential gravitational energy $\mathcal{E}_g$ can be rewritten as the sum of gravitational energy stored within every element E_q of the hair strand:

$$\mathcal{E}_g = \rho S g \sum_{Q=1}^{N} \left(\int_0^{\ell_Q} \left[\boldsymbol{n}_{0,Q}(u) \left(L - S_{Q-1} - u \right) \right] du \right) \cdot \boldsymbol{e}_{vert}$$

After splitting the integral in two parts, and pulling the constant $(L - S_{Q-1})$ out of the integral, the equation becomes:

$$\mathcal{E}_g = \rho S g \sum_{Q=1}^{N} \left(\left(L - S_{Q-1} \right) \int_0^{\ell_Q} \left[\boldsymbol{n}_{0,Q}(u) \right] du - \int_0^{\ell_Q} \left[u\, \boldsymbol{n}_{0,Q}(u) \right] du \right) \cdot \boldsymbol{e}_{vert}$$

The integral of $\boldsymbol{n}_{0,Q}(u)$ is defined to be $\boldsymbol{r}_Q(u)$, thus:

$$\mathcal{E}_g = \rho S g \sum_{Q=1}^{N} \left(\left(L - S_{Q-1} \right) \left[\boldsymbol{r}_Q(u) \right]_0^{\ell_Q} - \int_0^{\ell_Q} \left[u\, \boldsymbol{n}_{0,Q}(u) \right] du \right) \cdot \boldsymbol{e}_{vert}$$

which becomes:

$$\mathcal{E}_g = \rho S g \sum_{Q=1}^{N} \left(\left(L - S_{Q-1} \right) \left(\boldsymbol{r}_Q(\ell_Q) - \boldsymbol{r}_Q(0) \right) - \underbrace{\int_0^{\ell_Q} \left[u\, \boldsymbol{n}_{0,Q}(u) \right] du}_{\boldsymbol{A}} \right) \cdot \boldsymbol{e}_{vert}$$

Working out the integral $\boldsymbol{A}$:

$$\boldsymbol{A} = \int_0^{\ell_Q} \left[\boldsymbol{n}_{i,Q}^{\parallel}(0)\, u \; + \boldsymbol{n}_{i,Q}^{\perp}(0)\; cos(\Omega\, u)\, u + \boldsymbol{\omega} \times \boldsymbol{n}_{i,Q}^{\perp}(0)\; sin(\Omega\, u)\, u \right] du$$

which becomes:

$$\boldsymbol{A} = \left[\frac{\boldsymbol{n}_{0,L}^{Q\parallel}\, u^2}{2} + \boldsymbol{n}_{0,L}^{Q\perp} \left(\frac{\Omega\, u\; \sin(\Omega\, u) + \cos(\Omega\, u)}{\Omega^2} \right) + \boldsymbol{\omega} \times \boldsymbol{n}_{0,L}^{Q\perp} \left(\frac{\sin(\Omega\, u) - \Omega\, u\; \cos(\Omega\, u)}{\Omega^2} \right) \right]_0^{\ell_Q}$$

Which result in an implementable gravitational energy equation:

$$\mathcal{E}_g = \rho S g \sum_{Q=1}^{N} \left(\left(L - S_{Q-1} \right) \left(\boldsymbol{r}_Q(\ell_Q) - \boldsymbol{r}_Q(0) \right) - \boldsymbol{A} \right) \cdot \boldsymbol{e}_{vert}$$

where:

$$\boldsymbol{A} = \left(\frac{\boldsymbol{n}_{0,L}^{Q\parallel}\, {\ell_Q}^2}{2} + \boldsymbol{n}_{0,L}^{Q\perp} \left(\frac{\Omega \ell_Q \sin(\Omega \ell_Q) + \cos(\Omega \ell_Q) - 1}{\Omega^2} \right) + \boldsymbol{\omega} \times \boldsymbol{n}_{0,L}^{Q\perp} \left(\frac{\sin(\Omega \ell_Q) - \Omega \ell_Q \cos(\Omega \ell_Q)}{\Omega^2} \right) \right)$$

This equation calculates the vertical area $\int_0^L z(s)$ of the hair – where $z(0) = 0$. Multiplying this area with the hair density and the hair cross section results in finding the vertical centre of mass of the hair. A hair strand located mostly under $\boldsymbol{r}(0)$ would have a negative centre of mass, and a hair

strand located mostly above $\boldsymbol{r}(0)$ would have a positive centre of mass. Multiplying the centre of mass with the gravity field value results in the potential gravitational energy of the hair.

(Bertails, et al., 2005b), (Bertails, 2006)

4.1.1.4. Super-Helix Algorithm

At this point, the equations for calculating the centreline $\boldsymbol{r}(s)$ and the material frame $\boldsymbol{n}_i(s)$ of a Super-Helix and the equations for calculating the potential energy $\mathcal{E}_{hair}$ stored within the Super-Helix are known. These equations can be used to calculate the equilibrium shape of an individual hair strand subject to gravity and internal elasticity.

The first step in calculating the shape of a hair strand is by specifying how the hair strand is discretized; the rod will be divided into N elements with length ℓ_Q specified for every element Q. $\boldsymbol{\kappa}$**,** the vector of size $3N$ collecting the piecewise constant twist and curvatures $\kappa_{i,Q}$, $i = 0,1,2$, is initialized with the intrinsic natural torsion and curvatures τ^0, κ_1^0 and κ_2^0 of the hair strand:

$$\forall Q \;\; \kappa_{i,Q} = \kappa_{\mathrm{i}}^0$$

Under the assumption of the following redundant notation:

$$\kappa_0^0 = \tau^0$$

Natural quantities for the intrinsic torsion and curvature can be found in Table 2.

The argument for initialize $\boldsymbol{\kappa}$ with the natural torsion and curvature is that these would be the values that the vector would have when belonging to a hair strand whit no external forces applied to it. As the aim of the algorithm is to find the $\boldsymbol{\kappa}$ that minimizes the potential energy $\mathcal{E}_{hair}$, and computing the corresponding configuration of the rod, starting with the intrinsic torsion and curvatures would be an intuitive choice, as this would set the internal elastic energy $\mathcal{E}_e$ to the minimum value.

For minimization, energy $\mathcal{E}_{hair}$ will be initialised with the maximum value possible and until $\mathcal{E}_{hair}$ converges to a fixed value, the algorithm iteratively proceeds with the following steps:

- Compute the elastic energy $\mathcal{E}_e$ using the equation for internal elastic energy. The generalized coordinates $\boldsymbol{\kappa}$ act as input for the equations. The constants of the equation for internal elastic energy are user specified; natural quantities for the constants can be found in Table 1.
- Calculate formally the configuration of the rod, by recursively calculating the vectors $\boldsymbol{n}_{i,Q}(0)$ and $\boldsymbol{r}_Q(0)$ for every element E_Q. The generalized coordinates $\boldsymbol{\kappa}$**,** and the clamped position and orientation $\boldsymbol{r}_c$ and $\boldsymbol{n}_{i,c}$ act as input for the equations.
- Compute the gravitational energy $\mathcal{E}_g$ using the equation for potential gravitational energy. The generalized coordinates $\boldsymbol{\kappa}$ and the vectors $\boldsymbol{n}_{i,Q}(0)$ and $\boldsymbol{r}_Q(0)$ act as input for the equations. The constants of the equation for potential gravitational energy are user specified; natural quantities for the constants can be found in Table 3.
- Perform a minimization step for $\mathcal{E}_{hair} = \mathcal{E}_g + \mathcal{E}_e$, using the zero order gradient search approach (Rosenbrock, 1960). This search algorithm changes the values in vector $\boldsymbol{\kappa}$ to the

new location in parametric space that needs to be sampled for its energy $\mathcal{E}_{hair}$. The gradient search algorithm will be explained later in section ***Energy Minimization***.

Knowing the vector $\boldsymbol{\kappa}$ that minimizes energy $\mathcal{E}_{hair}$, and the position and orientation of the clamped end of the hair strand, the final shape of the hair strand is recursively calculated.

4.1.2. Hair Collision

To create realistic hair, it is necessary to account for both hair-object collision and hair self-collisions. Hair-object collisions are needed to prevent that hair strands penetrate objects in the scene, while hair self-collision is essential for giving an adequate volume of hair. The hair-object collision and hair self-collisions are computed with the help of primitive collision geometry, which act as approximations for the objects present in the scene.

Positions are sampled at fixed intervals Δs along the curvilinear abscissa of the hair strand. Every sampled position p_i – where $0 \leq i < \sum_Q (\ell_Q / \Delta s)$ – is then tested against the collision geometry for the amount of penetration. The collision response, which the collision geometry g_j has on position p_i, is computed using an elastic penalty force $F_{i,j}$, where the elastic penalty force is calculated with the following equation:

$$F_{i,j} = \frac{1}{2} k_j \left(\mathbf{max}(dent_{i,j}, 0) \right)^2$$

Above equation is derived from Hooke's spring equation; k_j is a user defined variable representing the stiffness parameter of the collision geometry g_j, and $dent_{i,j}$ represents the amount of penetration of position p_i within collision geometry g_j. To prevent a penalty force when the amount of penetration is negative, the minimum value of penetration is set to *zero*.

Summing all elastic penalty forces – for every sampled position p_i on the curvilinear abscissa, with every geometry g_j present in the scene – together results in the total amount of collision penalty that the hair configuration gathers. Because this energy should not be dependent on the amount of discretization, the sum is normalized by multiplying the sum with the step size Δs. The collision energy becomes:

$$\mathcal{E}_c = \sum_i \left(\sum_j (F_{i,j}) \right) \Delta s$$

As the Super-Helix model is based on energy minimization, accounting for collision detection and response simply amounts to minimizing the new energy $\mathcal{E}_{hair}$ defined as:

$$\mathcal{E}_{hair} = \mathcal{E}_g + \mathcal{E}_e + \mathcal{E}_c$$

Minimizing the value for $\mathcal{E}_{hair}$ will now result in a shape of the hair strand that tries to avoid penetration of collision geometry.

Note however that a small increase in collision energy could lead to a larger decrease in gravitational and elastic energy, resulting in a hair with less potential energy. This increase in collision energy indicates that the hair lies slightly beneath the surface of the collision geometry. To avoid that this

hair penetrates an object, it is advised to approximate this object with a collision geometry that is slightly larger than the object.

4.1.2.1. Collision Geometry

The collision geometry is responsible for calculating the penalty force $F_{i,j}$. The geometry is created by setting the centre, the orientation, the dimensions, and the stiffness parameter of the geometry.

Different geometrical shapes have been implemented, where every geometry contains a function that accepts a position in space as parameter and returns the corresponding penalty energy of the geometry.

Three geometric shapes have been implanted. These are the ellipsoid, the sphere and the cuboid. An ellipsoid is a good approximation to represent the collision hull of round objects. As an ellipsoid resembles the shape of a head, I have used this geometry to account for the hair-head collisions. The sphere – a special case of the ellipsoid, where all dimensions are equal – is a perfect approximating for the collision between hair and spherical objects. And last, the cuboid is implemented to account for collisions between hair and flat or box-like objects.

Although these three shapes are sufficient to account for most of the collisions in a scene, more complex shapes can be implemented and added to the hair simulation & rendering framework to approximate models with a higher complexity. The only criterion is that the model must implement an abstract collision interface, which specifies the method that calculates the penalty energy from a three-dimensional position in space.

In the following two sections I will describe how the penalty energy is calculated from the ellipsoid and cuboid. The sphere is explained together with the ellipsoid, as it is a special case of the ellipsoid shape.

4.1.2.1.1. Ellipsoid

In geometry, an ellipsoid is a type of shape that is a higher dimensional analogue of an ellipse. The equation of a standard axis-aligned ellipsoid body centred in a three-dimensional Cartesian coordinate system is:

$$\sqrt{\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2} \leq 1$$

where a and b are the equatorial radii along the $\boldsymbol{X}$-and $\boldsymbol{Y}$-axes respectively and c is the polar radius along the $\boldsymbol{Z}$-axis, all of which are fixed positive real numbers.

Figure 21 is an illustration of an ellipse – the two-dimensional analogue of the ellipsoid. This illustration will be used as reference to describe how the penetration of an ellipsoid is calculated. In the illustration $\boldsymbol{U}$ and $\boldsymbol{V}$ are unit vectors that represent the two axis of the local coordinate system. The ellipsoid will have a third unit vector $\boldsymbol{W}$. Specifying the directions of these vectors will determine the orientation of the ellipsoid. a and b specify the dimension of the ellipse. The ellipsoid will have a third dimension c. $\boldsymbol{P}$ represents the sampled positions, tested for penetration.

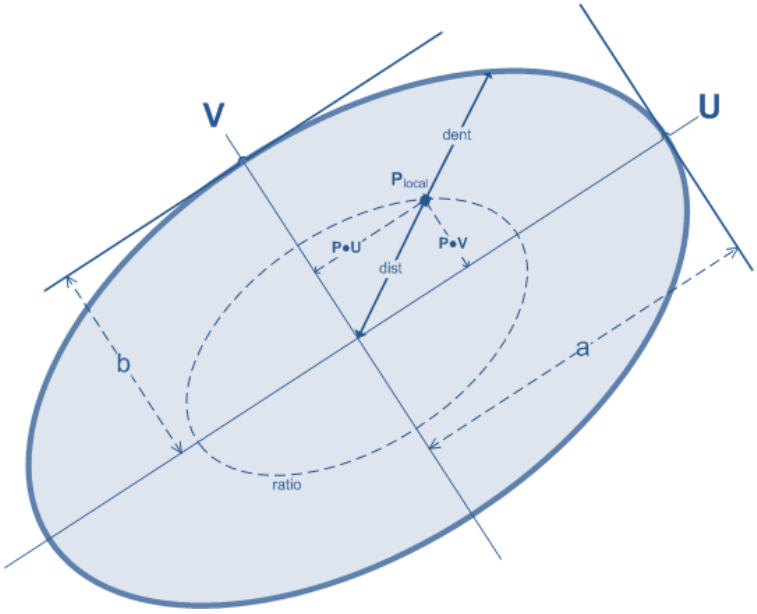

Figure 21. Penetration of an Ellipse

To calculate the amount of penetration that a position has with respect to an ellipsoid, the sampled position $\boldsymbol{P}_{world}$ is transformed into a vector $\boldsymbol{P}_{vector}$ which points from centre $\boldsymbol{C}$ of the collision geometry in the direction of position $\boldsymbol{P}_{world}$:

$$\boldsymbol{P}_{vector} = \boldsymbol{P}_{world} - \boldsymbol{C}$$

where the distance of the vector becomes:

$$dist = \|\boldsymbol{P}_{vector}\|$$

Projecting $\boldsymbol{P}_{vec}$ on each of the axis of the local coordinate system:

$$p_U = \boldsymbol{P}_{vector} \cdot \boldsymbol{U}$$

$$p_V = \boldsymbol{P}_{vector} \cdot \boldsymbol{V}$$

$$p_W = \boldsymbol{P}_{vector} \cdot \boldsymbol{W}$$

and writing $\boldsymbol{P}_{local}$ as:

$$\boldsymbol{P}_{local} = \langle p_U, p_V, p_W \rangle$$

Sampled position $\boldsymbol{P}_{world}$ has effectively been rewritten to a position $\boldsymbol{P}_{local}$, which is located in the local coordinate system of the collision geometry. Because this local coordinate system has its axis centred and aligned with the ellipsoid body, the equation of an ellipsoid body can be used for the positions located within this coordinate system.

Using the coordinates of $\boldsymbol{P}_{local}$ in the equation of an ellipsoid, the result becomes:

$$\sqrt{\left(\frac{p_U}{a}\right)^2 + \left(\frac{p_V}{b}\right)^2 + \left(\frac{p_W}{c}\right)^2} = ratio$$

where $ratio$ is a value greater or equal to *zero* representing the relative distance to the centre of the ellipsoid. A ratio of *zero* represents a position located at the centre of the ellipsoid, while a ratio of *one* represents a position located on the surface. A ratio greater than *one* indicates a position located outside the ellipsoid body.

When dividing $dist$ by $ratio$ we get the distance of the vector $\lambda \boldsymbol{P}_{vector}$**:**

$$\frac{dist}{ratio} = \|\lambda \boldsymbol{P}_{vector}\|$$

where λ is a positive value that scales $\boldsymbol{P}_{vector}$ so that the scaled vector becomes a vector pointing at the surface of the ellipsoid. Subtracting the $dist$ from the magnitude of this vector will result in the amount of penetration of position $\boldsymbol{P}_{world}$ into the ellipsoid:

$$dent = \frac{dist}{ratio} - dist$$

Where $dent$ represents the distance between $\boldsymbol{P}_{vector}$ and $\lambda \boldsymbol{P}_{vector}$.

Finally, the penalty collision energy can be calculated with the following equation:

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

Where k is the stiffness parameter of the collision geometry.

4.1.2.1.2. Sphere

In geometry, the sphere is a special case of the ellipsoid where all dimensions are equal. By replacing a, b and c in the equation of the ellipsoid with r, we get the equation of a sphere:

$$\sqrt{\left(\frac{x}{r}\right)^2 + \left(\frac{y}{r}\right)^2 + \left(\frac{z}{r}\right)^2} \leq 1$$

Or, when moving r to the left side of the equation:

$$\sqrt{x^2 + y^2 + z^2} \leq \mathrm{r}$$

where r is the radius of the sphere, which is a fixed positive real number.

The reason to additionally implement the sphere as a collision geometry, instead of using the more generic ellipsoid with equal dimensions $a = b = c$, is that some optimizations can be done with the calculations of penetration. These optimizations can result in a considerable decrease in calculation time when the number of sampled positions p_i is large.

Figure 22 is an illustration of a circle – the two-dimensional analogue of the sphere. This illustration will be used as reference to describe how the penetration of a sphere is calculated. In the illustration $\boldsymbol{X}$ and $\boldsymbol{Y}$ are unit vectors that represent the two axes of the global coordinate system. The sphere will have a third unit vector $\boldsymbol{Z}$. The value r specifies the radius of the circle, respectively sphere. $\boldsymbol{P}$ represents the sampled positions, tested for penetration.

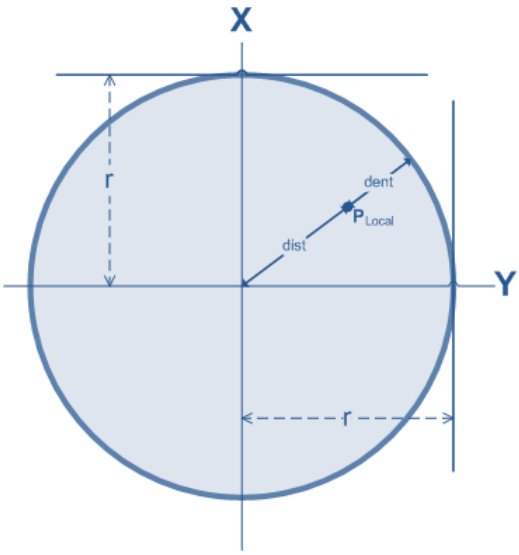

Figure 22. Penetration of a Circle

To calculate the amount of penetration that a position has with respect to a sphere, the sampled position $\boldsymbol{P}_{world}$ is transformed into a vector $\boldsymbol{P}_{vector}$ which points from centre $\boldsymbol{C}$ of the collision geometry in the direction of position $\boldsymbol{P}_{world}$:

$$\boldsymbol{P}_{vector} = \boldsymbol{P}_{world} - \boldsymbol{C}$$

where the distance of the vector becomes:

$$dist = \|\boldsymbol{P}_{local}\|$$

At this point, the calculations of the sphere become different from that of an ellipsoid. Because a sphere is point symmetric, it does not matter how the collision geometry is orientation of the local coordinate system. By assuming that the alignment of the local coordinate system matches that of the global coordinate system, we can use the x, y and z values of $\boldsymbol{P}_{vector}$ directly into the equation of the sphere without the need of projecting them on the axis of the local coordinate system:

$$\sqrt{(p_x)^2 + (p_y)^2 + (p_z)^2} = dist$$

where $dist$ is a value greater or equal to $zero$ representing the distance to the centre of the sphere. A ratio of $zero$ represents a position located at the centre of the sphere, while a ratio of r represents a position located on the surface. A ratio greater than r indicates a position located outside the sphere body.

Subtracting the $dist$ from the radius r of the sphere will result in the amount of penetration of position $\boldsymbol{P}_{world}$ into the sphere:

$$dent = r - dist$$

Where $dent$ represents the minimal distance between $\boldsymbol{P}_{vector}$ and the surface of the sphere.

Finally, the penalty collision energy can be calculated with the following equation:

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

Where k is the stiffness parameter of the collision geometry.

4.1.2.1.3. Cuboid

In geometry, a cuboid is a type of shape that is a higher dimensional analogue of a rectangle. The cuboid is a solid figure bounded by six rectangle faces, in which each pair of adjacent faces meets in a right angle. The equation of a standard axis-aligned cuboid body centred in a three-dimensional Cartesian coordinate system is:

$$\left|\frac{x}{a}\right| \leq 1 \ \wedge \ \left|\frac{y}{b}\right| \leq 1 \ \wedge \ \left|\frac{z}{c}\right| \leq 1$$

or:

$$|x| \leq a \ \wedge \ |y| \leq b \ \wedge \ |z| \leq c$$

where a is the distance of the left and right face along the $\boldsymbol{X}$-axis, b is the distance of the top and bottom face along the $\boldsymbol{Y}$-axis, and c is the distance of the front and back face along the $\boldsymbol{Z}$-axis. All of the distances are fixed positive real numbers.

Figure 21 is an illustration of a rectangle – the two-dimensional analogue of the cuboid. This illustration will be used as reference to describe how the penetration of the cuboid is calculated. In the illustration $\boldsymbol{U}$ and $\boldsymbol{V}$ are unit vectors that represent the two axis of the local coordinate system. The cuboid will have a third unit vector $\boldsymbol{W}$. Specifying the directions of these vectors will determine the orientation of the ellipsoid. a and b specify the dimension of the rectangle. The cuboid will have a third dimension c. $\boldsymbol{P}$ represents the sampled position, tested for penetration.

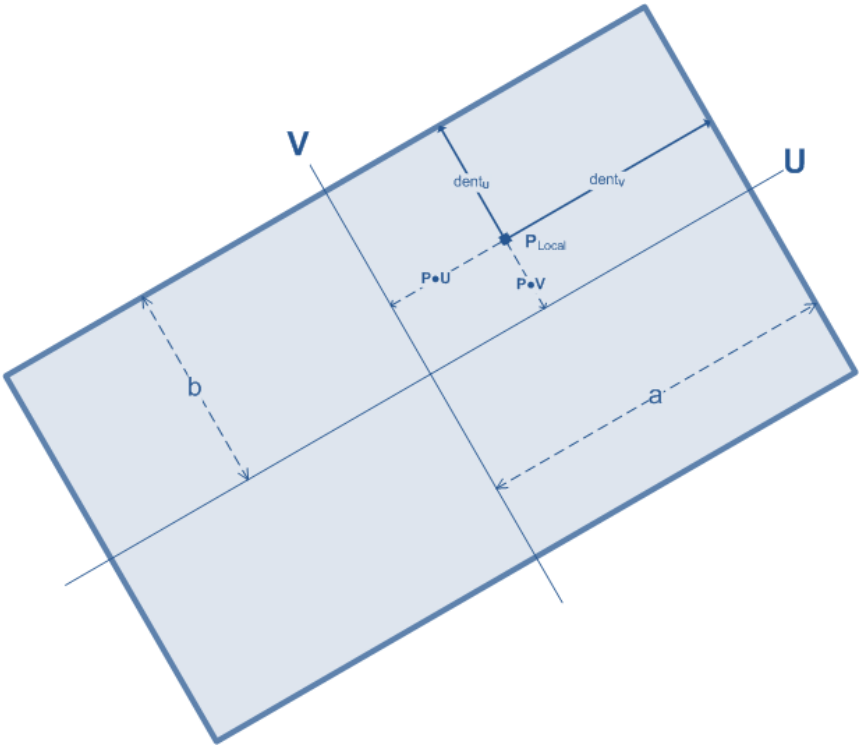

Figure 23. Penetration of a Rectangle

To calculate the amount of penetration that a position has with respect to a sphere, the sampled position $\boldsymbol{P}_{world}$ is transformed into a vector $\boldsymbol{P}_{vector}$ which points from centre $\boldsymbol{C}$ of the collision geometry in the direction of position $\boldsymbol{P}_{world}$:

$$\boldsymbol{P}_{vector} = \boldsymbol{P}_{world} - \boldsymbol{C}$$

Projecting $\boldsymbol{P}_{vec}$ on each of the axis of the local coordinate system:

$$p_U = \boldsymbol{P}_{vector} \cdot \boldsymbol{U}$$

$$p_V = \boldsymbol{P}_{vector} \cdot \boldsymbol{V}$$

$$p_W = \boldsymbol{P}_{vector} \cdot \boldsymbol{W}$$

and writing $\boldsymbol{P}_{local}$ as:

$$\boldsymbol{P}_{local} = \langle p_U, p_V, p_W \rangle$$

Sampled position $\boldsymbol{P}_{world}$ has effectively been rewritten to a position $\boldsymbol{P}_{local}$, which is located in the local coordinate system of the collision geometry. Because this local coordinate system has its axis centred and aligned with the ellipsoid body, the equation of an ellipsoid body can be used for the positions located within this coordinate system.

Using the coordinates of $\boldsymbol{P}_{local}$ in the equation of an ellipsoid, the result becomes:

$$|p_U| = dist_U \ \wedge \ |p_v| = dist_V \ \wedge \ |p_w| = dist_W$$

where $dist_U$, dis_V an $dist_W$ are values greater or equal to $zero$ representing the distance of $\boldsymbol{P}_{local}$ along the $\boldsymbol{U}$-, $\boldsymbol{V}$- and $\boldsymbol{W}$-axis of the local coordinate system respectively. A value of zero represent a position where the offset on the corresponding axis is *zero*, while a value that equals the value of the corresponding dimension represents a position on one of the two corresponding rectangular faces. A value greater than the value of the corresponding dimension represents a position located outside the cuboid body.

Because the values $dist_U$, dis_V an $dist_W$ are independent of each other – they do not influence each other; every value can be treated separately. Subtracting the distances from the dimensions a, b and c respectively will result in three different penetration values:

$$dent_U = a - dist_U$$

$$dent_V = b - dist_V$$

$$dent_W = c - dist_W$$

each corresponding to the minimum penetration of two opposite faces.

Finding the minimum value of the three penetrations will result in the amount of penetration of position $\boldsymbol{P}_{world}$ into the cuboid:

$$dent = \mathbf{min}(\langle dent_U, dent_V, dent_W \rangle)$$

Where $dent$ represents the minimal distance between $\boldsymbol{P}_{vector}$ and the nearest surface of the cuboid.

Finally, the penalty collision energy can be calculated with the following equation:

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

Where k is the stiffness parameter of the collision geometry.

4.1.2.2. Multiple Layered Envelopes

When minimizing the potential energy of a hair strand, a hair that would otherwise fall through a collision geometry, will now be pushed to a location near the surface of the corresponding collision geometry. In the case of the collision geometry of the head, the aggregated hair will be pushed to the surface of the geometry to form a thin sheath surrounding the head of the model. The hairstyle of the model will look surrealistic because it lacks the volume of natural hair. The volume of natural hair is caused by hair self-collision and should be approximated in the framework to create believable render results.

The collision detection between hairs is a very expensive operation. (Lee, et al., 2000) has developed a technique that adds volume to hair without worrying about the exact collision occurring between individual hair strands. The principle is based on the following hypothesis: the hair growing on the top of the head should always cover the hair growing below. The idea then is to create a series of envelopes around the head, and detecting the collision of each hair with the envelope corresponding to the elevation of the hair strand. This method is simply the extension of the algorithm for detecting collisions between the hair and the head. Of course, it only works if the head remains static and oriented vertically. A similar approach is taken in this project, and will be explained below.

Figure 24 is an illustration of the technique implemented in this project. This illustration shows an approximation of a head, represented by the filled ellipse, and a layered collision hull, represented by the area between the two dotted ellipses. ***P*** represents the sampled position, tested for penetration.

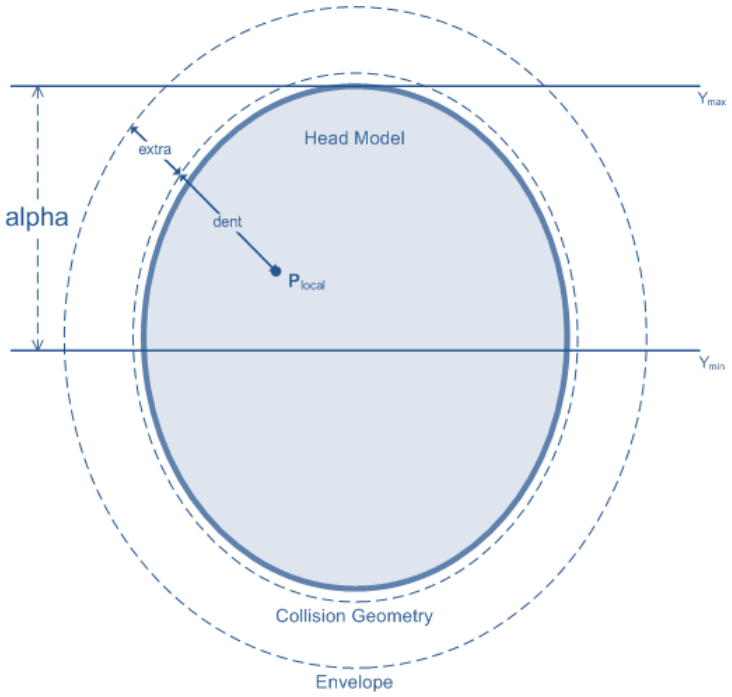

Figure 24. Layered Collision Envelopes

Originally, the collision geometry is created by setting the centre, the orientation, the dimensions, and the stiffness parameter of the geometry. Adding an additional value v – the volume – to the collision geometry, allows the calculation of a series of envelopes, and detecting the collision of each hair with the envelope corresponding to the elevation of the hair strand.

An envelope can be created by increasing the size of a collision geometry by a fixed amount $extra$ in all directions. Calculating the amount of penetration that a point makes with an envelope is thus the amount of penetration that a point makes with a collision geometry and adding $extra$ to the calculated amount of penetration:

$$dent_{envelope} = dent + extra$$

The value $extra$ is specified by the elevation of the hair follicle. Every hair has one end of the hair strand, $s = 0$, clamped into the head while the other end, $s = L$, is free of constrains. The clamped position $\boldsymbol{r}(0) = \boldsymbol{r}_c$ is the position where the hair follicle is located. By evaluating the position of every hair follicle, one can find the follicle with the lowest position and the follicle with the highest position on the head. Let y_{min} be the vertical elevation of the lowest follicle, let y_{max} be the vertical elevation of the highest follicle, and let $y_{current}$ be the vertical elevation of the follicle corresponding to the currently sampled hair. With this information $extra$ can be calculated by the following equation:

$$extra = alpha * v$$

where $alpha \in [0, 1]$ is the value of a linear interpolation between y_{min} and y_{max}. The value of $alpha$ is calculated with the following equation:

$$alpha = \frac{y_{current} - y_{min}}{y_{max} - y_{min}}$$

In the situation where $y_{current}$ equals y_{min}, the value of $extra$ will be zero. When $y_{current}$ equals y_{max}, the value of $extra$ will be v. For all other values of $y_{current}$, the value of $extra$ will range between zero and v.

The ellipsoid that approximates the head of the model has a positive v. This ensures that the hair strand growing on the top of the head should always cover the hair growing below. A negative v can be used for collision geometry located on the outer side of the hairstyle. In this case the hair growing below should always cover the hair strand growing on the top of the head. The magnitude of v determines the spacing between the hair strands, and thus the volume of the hairstyle.

The penalty collision energy of the envelope can be calculated with the following equation:

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

This equation will replace the collision energy calculations mentioned in the previous section. Note that resetting a possible negative amount of penetrations to *zero* happens after adding $extra$ to $dent$; a sampled position which is located outside the collision geometry can cause a penetration inside of the envelope.

4.1.3. Energy Minimization

In the paper of (Bertails, et al., 2005b) the energy minimization of the Cosserat Rods is performed by the Davidon–Fletcher–Powell method described in (Fletcher, et al., 1963). The Davidon-Fletcher-Powell optimization algorithm belongs to the family of first order optimization algorithms known as gradient descent or steepest-descent, which means that the algorithm finds a local minimum of a given function by iteratively traversing the parameter space with the path that has the steepest gradient. This process is continued until the gradient converges to zero.

While the Davidon-Fletcher-Powel algorithm is performing well in (Bertails, et al., 2005b), I have taken another approach in this project. I have chosen to use the Rosenbrock method instead. The Rosenbrock method, explained in (Rosenbrock, 1960), is a zero order search algorithm, which means that it does not require any derivatives or gradients from the basic function that needs to be optimized. While only simple evaluations of the basic function are required, it can approximate a gradient search; thus combining advantages of zero order and first order strategies. My opinion is that this algorithm is particularly well suited for the Cosserat Rod, because the derivatives of the energy function of the Cosserat Rod are more complex to calculate than the original energy functions; while more evaluations are needed with the Rosenbrock method, it will lead, in the long end, to a more stable method due to how the derivative of a function is approximated.

4.1.3.1. Optimization Introduction

The simplest way of finding a minimum of a function $f(\boldsymbol{x})$ would be to change the variables $\langle x_1, x_2, \cdots, x_n \rangle$ in turn, reducing f as far as possible with each variable and then passing on to the next. With two parameters, the method works well if the contour lines of the plotted f are nearly circular, but becomes very slow when there exist a correlation between the two dimensions. The existence of an interaction corresponds to the presence of a long, narrow ridge when plotting the function. If this ridge is not parallel to one of the two axes, progress will only be made by small steps in x_1 and x_2 in turn.

Because of this difficulty, the method of steepest descent is usually recommended. In its general form, this approach consists in finding the direction of steepest descent from:

$$\xi_i = \frac{\partial f}{\partial x_i} \Big/ \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i}\right)^2}$$

Where ξ_i are the components of a unit vector $\boldsymbol{\xi}^0$ in the required direction. The value of f is then calculated at new points along a line from the first point parallel to $\boldsymbol{\xi}^0$ until the least value is attained. Starting from this lowest point, the process is repeated; the values for ξ_i are evaluated again and progress is made along a line parallel to the new vector $\boldsymbol{\xi}^1$.

At first sight this method seems very effective, but it has a big disadvantage. The vector $\boldsymbol{\xi}^0$ will be perpendicular to the contour at the initialisation point and progress will be made until the local contour is parallel to $\boldsymbol{\xi}^0$. At this point $\boldsymbol{\xi}^1$ will be found, and it is perpendicular to the local contour, and therefore to $\boldsymbol{\xi}^0$. Similarly $\boldsymbol{\xi}^2$ will be perpendicular to $\boldsymbol{\xi}^1$ and hence parallel to $\boldsymbol{\xi}^0$. Thus with two variables the method of steepest descent is equivalent to the method of changing one parameter at a time.

The two directions which will be used are fixed once for all by the choice of the initialisation point, and need have no relation to the direction of any ridges that may be present. If the contours are nearly circular, the method of steepest descent will have a small advantage, but this situation has often no practical importance.

Figure 25 is an illustration that shows the behaviour of steepest-descent on the Rosenbrock function. The Rosenbrock function is a non-convex function often used as a performance test problem for optimization algorithms. The Rosenbrock function is also known as Rosenbrock's valley or Rosenbrock's banana function, as the global minimum is inside a long, narrow, parabolic shaped flat valley. In the illustration it becomes clear that steepest-decent has no problem with finding the valley, but needs many small perpendicular steps to converge to the global minimum.

The Rosenbrock function is defined by:

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - {x_1}^2)^2$$

And has its global minimum at $f(1, 1) = 0$.

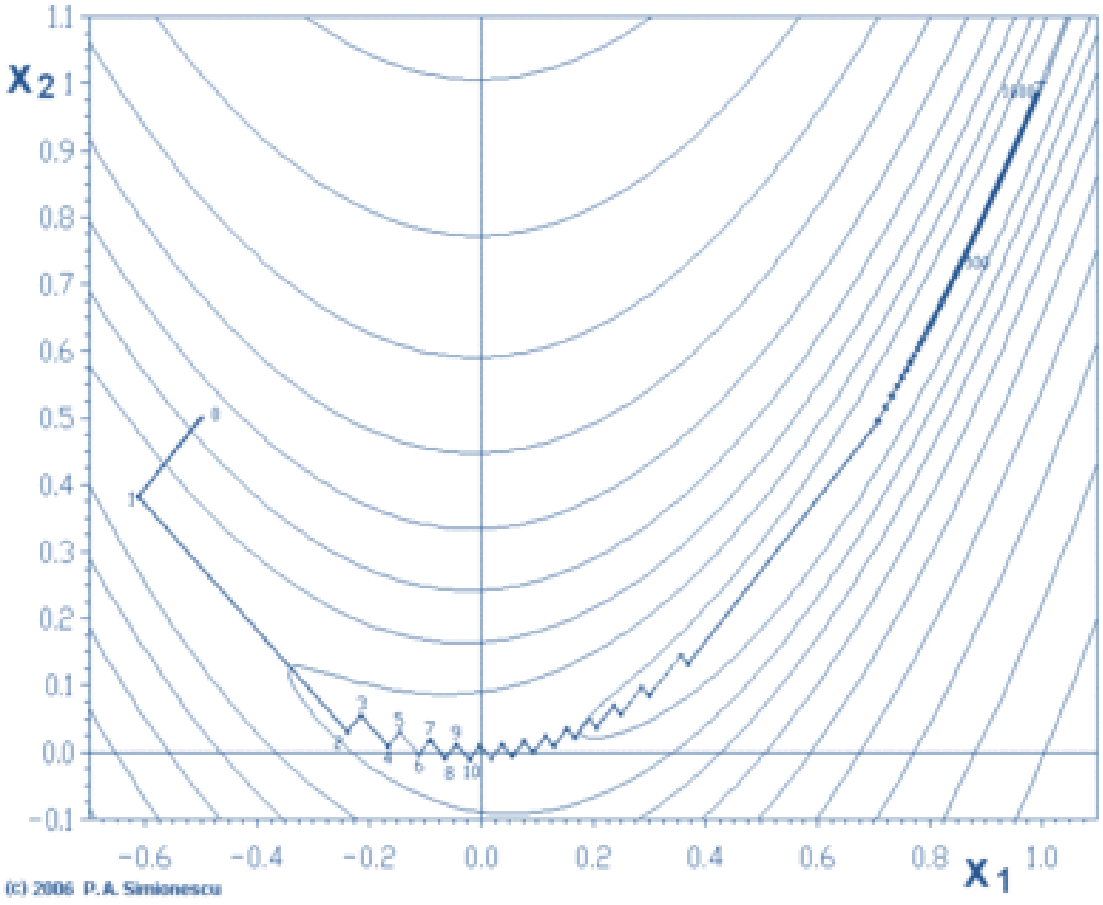

Figure 25. Rosenbrock Function

When there are more than two variables, the two methods are no longer equivalent: each vector ξ in the method of steepest descent is normal to the preceding vector, but n successive vectors do not necessarily form a mutually orthogonal set. However, (Rosenbrock, 1960) believes that the method of steepest descent will have no advantage compared to the method of changing one parameter at a time; certainly not if the main difficulty in a problem arises from the interaction of only two variables.

To overcome this disadvantage, (Booth, 1955) has proposed a modification to the method of steepest descent. The direction $\boldsymbol{\xi}$ is found as before, and progress is made parallel to it until the least value is attained. The distance from the starting point is then multiplied by 0.9 and is evaluated at the point so found. After a fixed number of repetitions of this procedure one step of full length is taken. By only traversing a given direction partially, we break the condition that every new direction $\boldsymbol{\xi}_{i+1}$ will be perpendicular to the current direction $\boldsymbol{\xi}_i$ being traversed. A step of full length at every fixed number of repetitions is needed to guarantee that the local minimum will eventually be reached.

The modification proposed by (Booth, 1955) would solve the problem, but there is a further difficulty with steepest descent. The partial derivatives of f must be obtained numerically at the beginning of each step. This can be done by evaluating f at two neighbouring points and using for example:

$$\frac{\partial f}{\partial x_1} \sim \frac{f(x_1 + h, x_2, \cdots, x_n) - f(x_1, x_2, \cdots, x_n)}{h}$$

when approximating the partial derivative of x_1. The value of h in above equation must be small enough so that the second term of the tailor polynomial:

$$\frac{h^2}{2}\frac{\partial^2 f}{\partial {x_1}^2}$$

can be neglected. However when h is as small as this, the method will fail to detect small values of $\partial f / \partial x_1$ due to a lack of precision in the data format used for calculation. This can lead to problems at plateaus of a function f, where the optimization method fails to detect the correct direction $\boldsymbol{\xi}$ of the next step. Other optimization problems that numerically calculate their gradient exhibit the same problem, including Davidon–Fletcher–Powell minimization method used in (Bertails, et al., 2005b).

To tackle above problems, (Rosenbrock, 1960) developed a new method that is better suited for numerical calculating the optimum of a function. This method, called the Rosenbrock method, will be explained in the next section.

4.1.3.2. Rosenbrock Method

(Rosenbrock, 1960) identifies three main problems which have to be met in developing a practical optimization method. These three problems are:

- Determining the length of a step
- Determining the direction of a step
- Inserting the limits and constraints of the function f

Because the generalized coordinates $\boldsymbol{\kappa}$ – which contain the curvature and torsion values of the Super-Helix – are free of any constraints, the problem of inserting the limits and constraints of a function can be discarded from this project. The solution of (Rosenbrock, 1960) to the other two problems will be described below.

The first problem is to decide the length of a step to be taken in the desired direction, assuming this direction is known. The approach taken by (Rosenbrock, 1960) is to try a step of arbitrary length ℓ. If

this step succeeds, ℓ is increased by multiplying it with α – where $\alpha > 1$. If it fails, ℓ turns sign and decreases by multiplying it with $-\beta$, where $0 < \beta < 1$. An attempt is Successful if the new found value of f is less than or equal to the previous value. Thus if ℓ is initially so small that it makes no change in the function f, it will be increased on the next attempt. Each such attempt is called a trial.

The second problem is to decide when and how to change the directions $\boldsymbol{\xi}$ in which the steps of length ℓ are taken. The approach taken by (Rosenbrock, 1960) is to work throughout n orthogonal directions $\boldsymbol{\xi}_i$ – where $1 < i < n$ – rather than choosing a single direction in which to progress at each stage. The reason to do this is that it is necessary anyway to examine the neighbouring points in each of n directions, in order to determine the best direction of advance. If one of the points examined in this way makes f less than the previous value, it could as well be accepted as a new starting point.

Trials are performed in each of the n orthogonal directions in turn, and this process continues until at least one trial has been successful in each direction, and one has failed thereafter. A trial in a given direction will eventually succeed because, after repeated failures, ℓ becomes so small that it causes no change in the function f. The set of trials made with one set of orthogonal directions, and the subsequent change of these directions are called a stage.

After the completion of a stage a new set of orthogonal direction $\boldsymbol{\xi}^1$ will be calculated, such that $\boldsymbol{\xi}_1^1$ lies along the direction of greatest advance for the previous stage, i.e. along the line joining the first and last points of that stage in n-dimensional space. For this purpose, (Rosenbrock, 1960) proposes the following calculating sequence:

$$\boldsymbol{A}_k = \sum_{i=k}^{n} \left(d_i \boldsymbol{\xi}_i^0 \right)$$

$$\boldsymbol{B}_k = \boldsymbol{A}_k - \sum_{j=1}^{k-1} \left(\left(\boldsymbol{A}_k \cdot \boldsymbol{\xi}_j^1 \right) \boldsymbol{\xi}_j^1 \right)$$

$$\boldsymbol{\xi}_k^1 = \boldsymbol{B}_k / |\boldsymbol{B}_k|$$

Where $\boldsymbol{\xi}^0$ contains the directions of the previous trial, d_i contains the sum of all successful steps ℓ_i in the direction of $\boldsymbol{\xi}_i^0$, and $\boldsymbol{\xi}^1$ contains the calculated directions for the next trial.

$\boldsymbol{A}_k$ are obtained by starting with the "greatest advance" vector $\boldsymbol{A}_1$, as defined above, and removing from it the successive orthogonal advance vector components $d_i \boldsymbol{\xi}_i^0$. The $\boldsymbol{B}_k$ are derived from the corresponding $\boldsymbol{A}_k$ by removing the components of $\boldsymbol{A}_k$ parallel to all the previously determined $\boldsymbol{\xi}_j^1$, so that the $\boldsymbol{B}_k$ are mutually orthogonal. Then by dividing each $\boldsymbol{B}_k$ by its magnitude, the corresponding unit vector $\boldsymbol{\xi}_k^1$ is obtained. Above calculating sequence is known as the Gram-Schmidt process, or Gram-Schmidt Orthonormalization, and is often used in situations where one needs to calculate the orthonormal vectors of a vector.

Figure 26 is an illustration showing an example of the Rosenbrock method performed on a two-dimensional function. In the first iteration, the Rosenbrock method is a simple search in the directions of the base vectors of the n-dimensional coordinate system. In the case of a success, which is an attempt yielding a new minimum value of the target function, the step width is increased, while in the case of a failure it is decreased and the opposite direction will be tried.

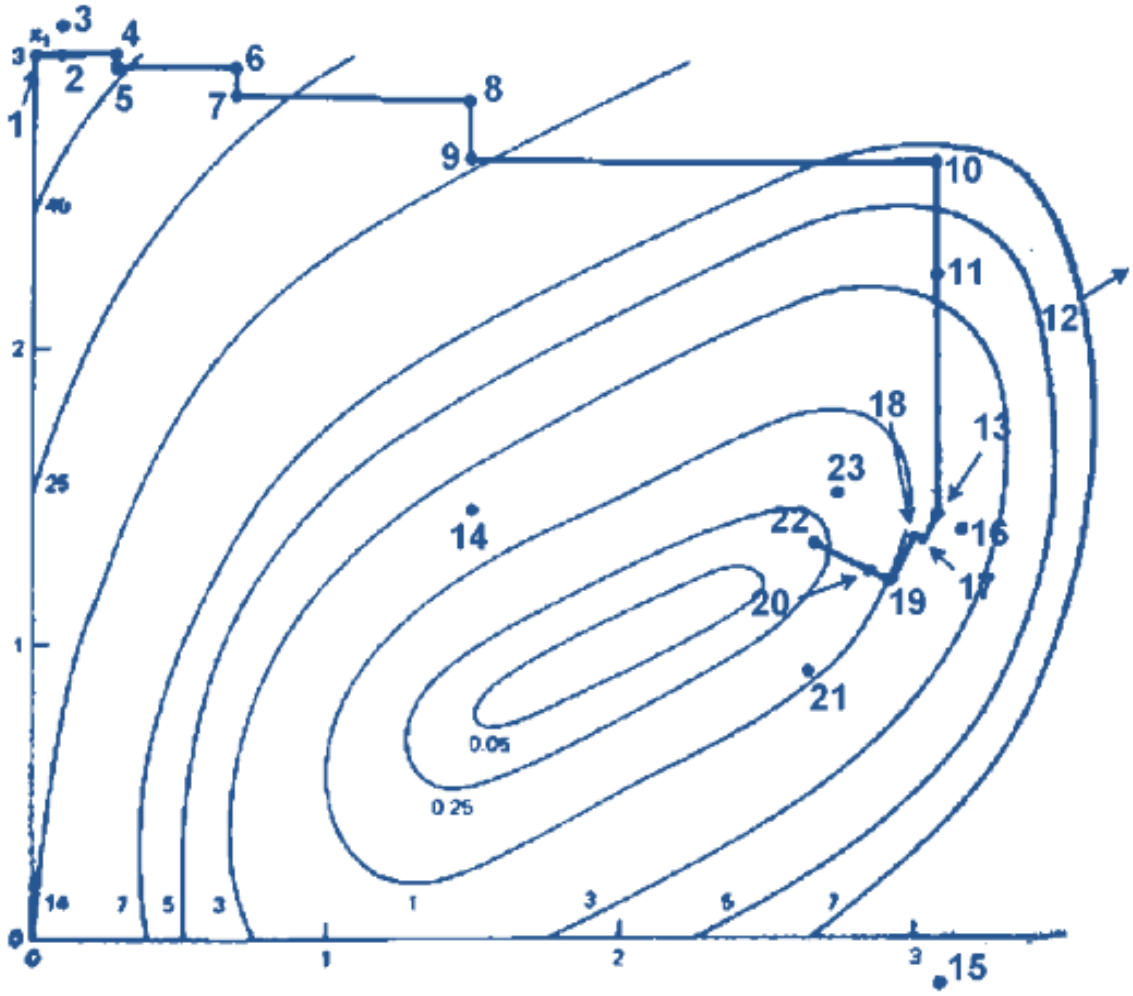

Figure 26. Rosenbrock Method

In the illustration, the Rosenbrock method is initialized at the location specified by 1. The value of the function f at this position is remembered and distances ξ_1 and ξ_2 are initialized to the x-axis and y-axis respectively.

A trial is performed along positive ξ_1 at a distance ℓ_1, resulting in location 2. Because the value of f at position 2 is less than or equal to the previously remembered value, the trial is considered a success and the value at this position replaces the previous value. Because the trial is a success, ℓ_1 is increased by multiplying it with α. The next trial is performed in the orthogonal direction along positive ξ_2 at distance ℓ_2, resulting in location 3. Because the value of f at position 3 is more than the remembered value, the trial is considered a failure and the algorithm returns to location 2. Because the trial is a failure, ℓ_2 decreases and changes sign by multiplying it with $-\beta$. This position continues until position 15 is reached. At this point both directions, ξ_1 and ξ_2, have met the condition of having at least one situation where a failed trial follows a number of successful trials; a stage has ended.

Once a success has been found and exploited in each direction, the coordinate system is rotated in order to make direction ξ_1 point into the direction of the gradient; ξ_1 becomes the unit vector that points from position 1 to position 13, and ξ_2 becomes the vector perpendicular to ξ_1. This rotation is achieved by the Gram-Schmidt process. Position 13, 16 and 17 are now defining the new

directions. Distances ℓ_1 and ℓ_2 are reset and the process of a stage is repeated using the rotated coordinate system.

The process of trials and stages are repeated until the function f stops decreasing, or until the maximum specified number of iterations is reached. The optimum parameters of a function are found by the Rosenbrock method by returning the vector $\boldsymbol{x}$ for which $f(\boldsymbol{x})$ is minimal.

Due to its simplicity, the Rosenbrock method is more stable than many sophisticated algorithms and it requires much less calculations of the target function than higher order strategies. The Rosenbrock method is easy to implement and is guaranteed to find a local minima of f if it finishes correctly and a minima is present.

4.1.3.3. Palmer Improvement

As mentioned in the previous section, the original Rosenbrock method of (Rosenbrock, 1960) calculates the new set of orthogonal directions $\boldsymbol{\xi}$ by using the Gram-Schmidt process. However, this procedure becomes numerically instable when any of the distances d_i becomes zero or when more of the distances d are so small that they are lost in the summations when calculating the vectors $\boldsymbol{A}_i$.

In the case that distance d_p – where $1 \leq p < n$ – becomes *zero*:

$$\boldsymbol{A}_p = \boldsymbol{A}_{p+1}$$

(Palmer, 1969) proofs that:

$$\boldsymbol{B}_{p+1} = \boldsymbol{0}$$

so that

$$\boldsymbol{\xi}_{p+1}^1 = \boldsymbol{B}_{p+1} / \left|\boldsymbol{B}_{p+1}\right|$$

is undetermined.

This instability can lead to a premature ending of the optimization algorithm, resulting into a failure of finding the minimum of a function. The Gram-Shmidt orthogonalization procedure can also become very time consuming when the dimension n of the search space increases, because the complexity of the procedure is n^3.

(Palmer, 1969) developed an improved procedure for generating the orthonormal directions $\boldsymbol{\xi}$ of the Rosenbrock method that should replace the Gram-Schmidt process. The new procedure has considerable savings in time and in storage requirements, and is stable with the cases in which the original method would fail.

(Palmer, 1969) identifies that if $\boldsymbol{B}_{p+1}$ and its magnitude $\left|\boldsymbol{B}_{p+1}\right|$ are evaluated that they prove to be proportional to distance d_p, so that in evaluating $\boldsymbol{\xi}_{p+1}^1$ the quantity d_p would cancel**,** leaving $\boldsymbol{\xi}_{p+1}^1$ determinate even if $d_p = 0$.

Working out $\boldsymbol{B}_k$ and $|\boldsymbol{B}_k|$ in terms of distances d_i and directions $\boldsymbol{\xi}_i^0$ showed that the vector $\boldsymbol{\xi}_k^1$ can be calculated by a recursive relation between A_k and A_{k-1}. The new equation for calculating direction $\boldsymbol{\xi}_k^1$ becomes:

$$\xi_k^1 = \frac{A_k|A_{k-1}|^2 - A_{k-1}|A_k|^2}{|A_{k-1}||A_k|\sqrt{(|A_{k-1}|^2 - |A_k|^2)}}$$

The full mathematic proof of ξ_k^1 can be found in (Palmer, 1969).

4.1.4. Hair Layout & Growth

Although the minimization of the hairs potential energy – determining the shape of a single hair strand – can be performed within a fraction of a second, the time begins to add up when one wants to calculate an entire hair coupe of a person with around 100 000 to 150 000 hair strands.

One solution, proposed by (Daldegan, et al., 1993), considering that the hair has an overall coherence in its appearance during the movement, one can simply animate only a few guide hairs, then expand this movement to the rest of the hair by interpolation. The disadvantage is that it only applies to very smooth hair, and, also, shows no discontinuity in the movement. In addition, this method poses the problem of collisions between the hair and body, since only a few hairs are able to detect interpenetration.

Another solution is to consider that hair has not an overall continuity, but rather has local internal cohesion. The hair is then modelled as a set of geometric primitives called wisps, which are animated separately. Obviously, this model also reduces the number of individual elements to animate, while keeping some complexity to the hair. Several studies have been based on this approach, some simulating a small number of coarse wisps, others using surfaces for wisps, such as (Koh, et al., 2001) and (Koh, et al., 2000), or volumes for wisps, such as (Noble, et al., 2004), these wisps can be deformable or rigid.

The model used in this project is a combination of the two solutions described above. To be able to handle both smooth and clumpy hairstyles, I avoid choosing between a continuum and a wisp-based representation for hair. Many real hairstyles display an intermediate behaviour with hair strands being more evenly spaced near the scalp than near the tips. My solution, based on (Bertails, et al., 2006) uses a semi-interpolating scheme to generate non-simulated hair strands from the guide strands; the framework ranges from full interpolation to generate the extra hair strands near the scalp to full extrapolation within a hair wisp near the tips. The separation strongly depends on the level of curliness: straight hair requires relatively more interpolation – compared to extrapolation – than curly and clumpy hair.

4.1.4.1. Wick and Follicles

Given the large number of hair strands composing human hair, it is unimaginable to manually specify the location of every single hair on a virtual character. To make the task of determining the hair layout feasible in a reasonable time, a virtual wick is imported into the framework which specifies the position and orientation of the hair follicles. These hair follicles will be used to grow the guide hairs, which will determine the overall shape of the hair coupe.

The wick is dedicated mesh geometry, invisible in the rendered image, which approximates the scalp of the virtual character. Every vertex element in this mesh geometry specifies the location of a single hair follicle aligned along the normal vector of that vertex. The advantage of having a dedicated mesh for determining the positions and alignments of the hair follicles is that the level of detail of the follicle mapping can be changed without affecting the visual character model.

Figure 19 contains two screenshots taken from within the hair framework. The left image shows a geometry mesh representing the virtual characters head, and the right image shows a wireframe mesh representing the wick of the virtual character; this wick will be invisible during rendering. The virtual character head model and the wick model are both modelled with the use of a commercial three-dimensional modelling tool.

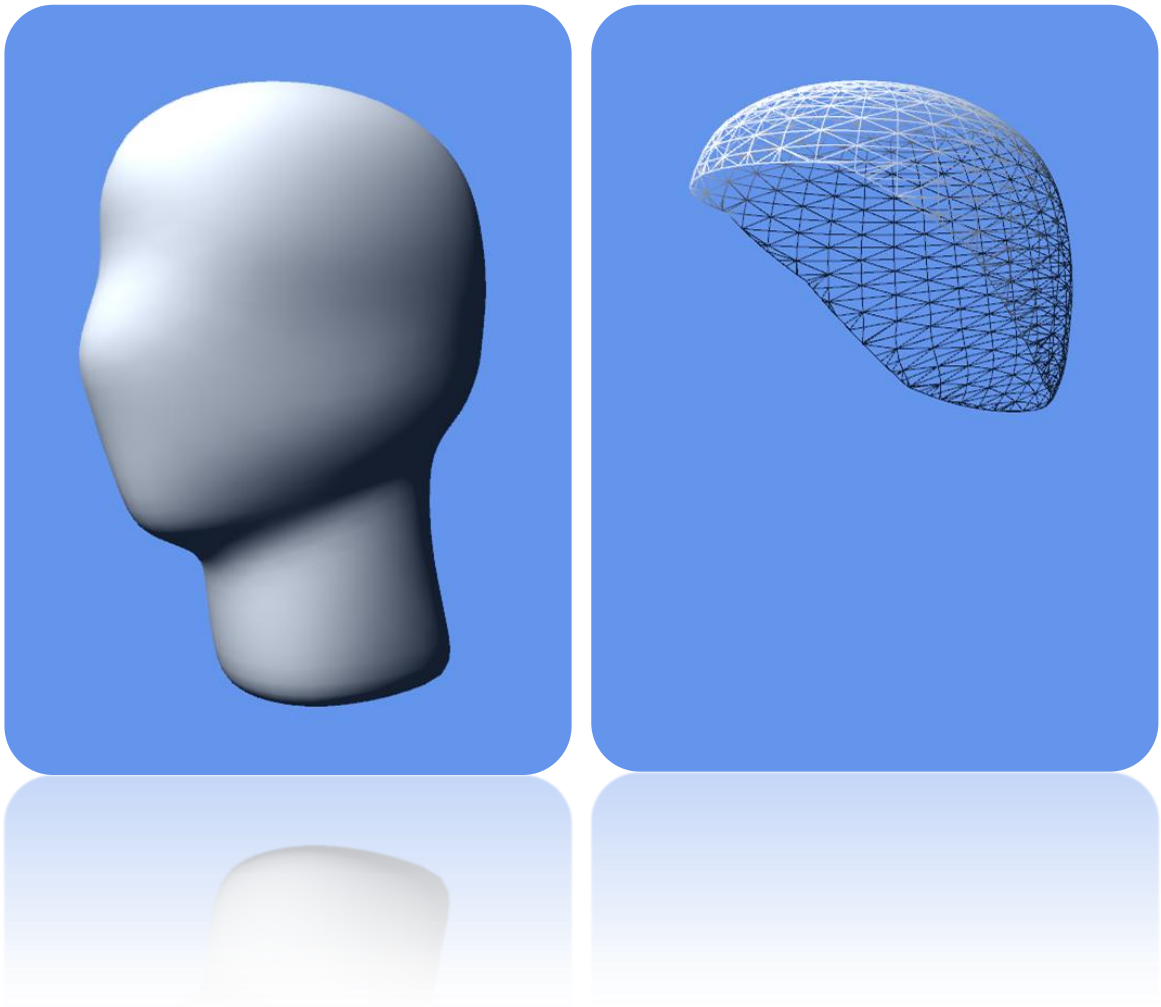

Figure 27. Head and Wick Model

To allow hair strands to grow from the head, appropriate boundary conditions need to be specified. As mentioned before in section ***Cosserat Rods***: one end of a hair strand is clamped into the head while the other end is free. The position of the clamped end $\boldsymbol{r}_c$, together with the orientation of the initial frame $\boldsymbol{n}_{i,c}$, are used as input in the equations needed for the potential energy minimization and the Super-Helix reconstruction process required to determine the shape of every guide strand.

The clamped position $\boldsymbol{r}_c$ and the tangent component $\boldsymbol{n}_{0,c}$ of the material frame are already specified by the position and alignment of the hair follicles. The other two orthonormal vectors $\boldsymbol{n}_{\alpha,c}$ – where $\alpha = 1,2$ – spanning the plane of the follicles cross section need to be calculated.

$\boldsymbol{n}_{1,c}$ can be obtained with the following equation:

$$\boldsymbol{n}_{1,c} = \frac{\boldsymbol{n}_{0,c} \times \boldsymbol{z}}{|\boldsymbol{n}_{0,c} \times \boldsymbol{z}|}$$

where $\boldsymbol{z} = \langle 0, 1, 0 \rangle$ is the vector pointing down along the vertical axis of the world. The cross product in above equation take the vectors $\boldsymbol{n}_{0,c}$ and $\boldsymbol{z}$ as input and results in a horizontal vector perpendicular to $\boldsymbol{n}_{0,c}$. Dividing this cross product by its magnitude ensures that the vector will be of unit length. In the two special cases, $\boldsymbol{n}_{0,c} = \boldsymbol{z}$ and $\boldsymbol{n}_{0,c} = -\boldsymbol{z}$, the cross product will result in a zero vector $\boldsymbol{0}$ and $n_{1,c}$ will be undetermined. In this situation, the vector $\boldsymbol{n}_{1,c}$ will be replaced with vector $\langle 0, 0, 1 \rangle$.

$\boldsymbol{n}_{2,c}$ can be obtained with the following equation:

$$\boldsymbol{n}_{2,c} = \boldsymbol{n}_{0,c} \times \boldsymbol{n}_{1,c}$$

Because both vectors in this cross product are of unit length, the resulting vector will be of unit length perpendicular to both $\boldsymbol{n}_{0,c}$ and $\boldsymbol{n}_{1,c}$.

4.1.4.2. Guide Hairs

The shape of the guide hairs are determined by using the clamped position and orientation of the corresponding hair follicles as input in the Super-Helix algorithm described in section 4.1.1.4. The Super-Helix algorithm determines for every hair strand h_j the vector $\boldsymbol{\kappa}$ containing the torsions and curvatures of the Super-Helix for which the energy $\mathcal{E}_{hair}$ is minimal. Remember that at this point the original hair energy calculation is replaced with an equation that includes the term representing the collision penalty with the corresponding collision envelope:

$$\mathcal{E}_{hair} = \mathcal{E}_g + \mathcal{E}_e + \mathcal{E}_c$$

When the minimum energy of a hair is found, the vector $\boldsymbol{\kappa}$**,** outputted by the minimization algorithm, can be used to reconstruct the functions $\boldsymbol{r}(s)$ and $\boldsymbol{n}_0(s)$, which respectively describe the centreline and the tangent along the curvilinear abscissa of the hair strand.

By sampling the functions $\boldsymbol{r}(s)$ and $\boldsymbol{n}_0(s)$ of a guide hair H_j – at fixed intervals Δs along the curvilinear abscissa – positions $p_{i,j}$ and tangents $t_{i,j}$ can be evaluated. The positions and tangents sampled in this way, are stored as vertex elements $G_{i,j}$ – where $G_{i,j} = \langle p_{i,j}, t_{i,j} \rangle$ – and form the discrete representation of the guide hair H_j.

4.1.4.3. Interpolated Hair

Once the shape of every guide hair H_j is discretized into a list of vertex elements $G_{i,j}$, the framework can start filling up the areas between the guide hairs with additional interpolated hairs. In this project is chosen for an evenly spaced distribution of the interpolated hairs near the roots and a clumping of the interpolated hairs, to the closest guide hair, near the tips. This behaviour corresponds fairly well to the distribution of natural human hair.

To fill up the area between guide hairs, a triangular face of the dedicated wick mesh is considered. Each triangular surface has three follicles growing guide hairs – one guide hair is grown at every vertex corner of the surface. To fill up the inside of the triangular surface with hairs, the framework will interpolate the sampled positions $p_{i,j}$ and tangents $t_{i,j}$ stored in the vertex elements of the three guide hairs to create the vertex elements of the newly interpolated hair. The interpolated hairs will have the same number of vertex elements as the three guide hairs.

Figure 28 is an illustration showing a triangular face with the corners specified by vertices A, B and C. Three guide hairs H_A, H_B and H_C are grown from the follicles located in the corners of the triangle. The inside area of the triangle surface is filled with additional hairs by interpolating the attributes of the sampled vertex elements $G_{i,A}$, $G_{i,B}$ and $G_{i,C}$ of the three guide hairs H_A, H_B and H_C.

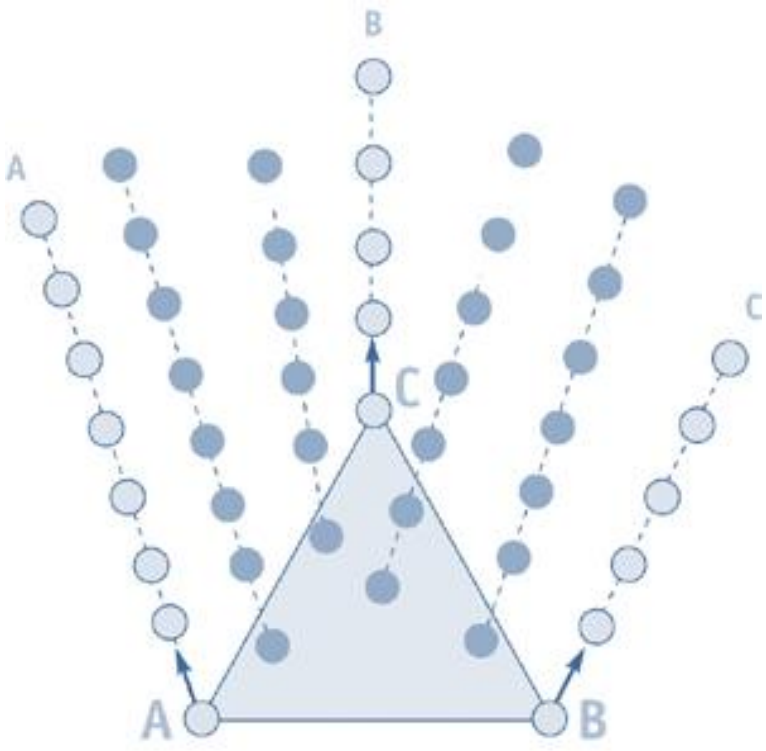

Figure 28. Hair Filling Process

To calculate the number of interpolated hairs N needed to fill up a triangular surface, the framework multiplies the triangular area with a user specified hair density. The density of natural hair will typically range between the 200 to 300 hair strands per square centimetre, but this value can be lowered by the user to increase the frame rate or decrease the memory requirements on lower end GPUs. The equation, for calculating the number of hair strands needed per triangle, becomes:

$$N = Area * Density$$

where $Density$ is a user specified hair density, and $Area$ specifies the surface area of the corresponding triangle. The value of $Area$ can be calculated easily due to the fact that the magnitude of a cross product can be interpreted as the positive area of the parallelogram having the two vectors of the cross product pointed along the sides of the parallelogram. This magnitude need to be divided by two, because the parallelogram has twice the surface area of the triangle. The equation for calculating $Area$ thus becomes:

$$Area = \frac{|(Pos_B - Pos_A) \times (Pos_C - Pos_A)|}{2}$$

where Pos_A, Pos_A and Pos_A are the positions of vertices A, B and C respectively.

Once the value of N is evaluated, the triangular surface area can be filled with N interpolated hairs. The vertex elements of an interpolated hair strand are determined by randomly generating

barycentric coordinates for that hair. The generated barycentric coordinates are used as weights for the vertex elements of the three guide hairs H_A, H_B and H_C in the interpolation process. This approach is partly based on NVidia's solution for the hair interpolation of the mermaid Nalu, which can be found in (Matt, 2005).

When specifying barycentric coordinates, the proportions of the three barycentric coordinates b_A, b_B and b_C must sum to some constant K. Usually, when the barycentric coordinates need to represent weights or proportions, this constant is specified as 1.0 or 100%. Because $b_A + b_B + b_C = K$ for all coordinates, any one variable is not independent from the other two, so only two variables need to be specified to find the remaining intersecting coordinate point in the triangular region. The following constraints must hold for the selection of the barycentric coordinates:

$$b_A = K - (b_B + b_C)$$

$$b_B = K - (b_C + b_A)$$

$$b_C = K - (b_A + b_B)$$

Figure 29 is an illustration showing a ternary plot which has A, B and C as its variables. The coordinates of this particular plot sum to constant $K = 1$. The ternary plot is drawn as a triangle, where each base, or side, of the triangle represents a weight of 0 or 0% , with the point of the triangle opposite that base representing a weight of 1 or 100% . As a weight increases in any one sample, the point representing that sample moves from the base to the opposite point of the triangle. Typical coordinate positions are listed in the illustration.

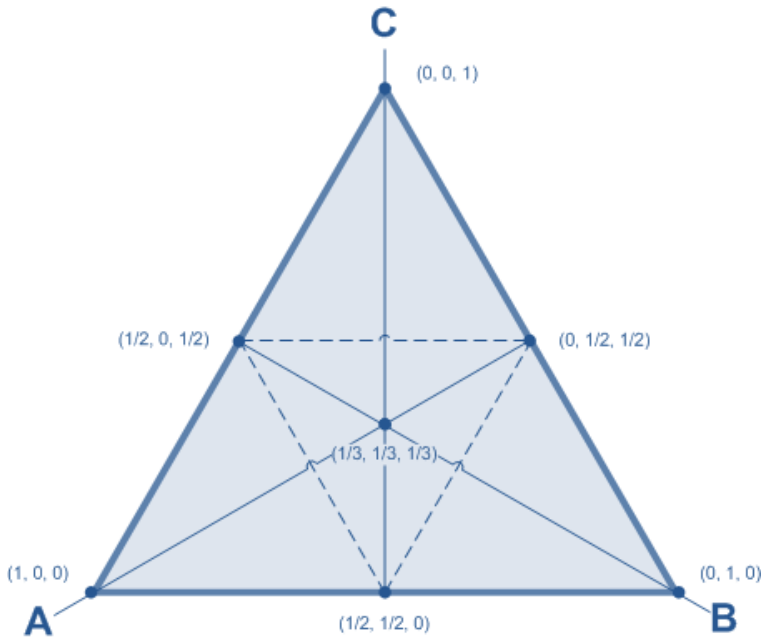

Figure 29. Barycentric Coordinates

The vertex elements of an interpolated hair H_Y can be determined by interpolating the vertex elements of the guide hairs H_A, H_B and H_C, where the barycentric coordinates act as weights for the corresponding vertex elements. The equation used in this project for calculating the vertex elements of the interpolated hair is as follows:

$$I_{i,Y} = G_{i,A}\, b_{i,A} + G_{i,B}\, b_{i,B} + G_{i,C}\, b_{i,C}$$

In above equation $I_{i,Y}$ is the ith vertex element of interpolated hair H_Y, and $b_{i,A}$, $b_{i,B}$ and $b_{i,C}$ are the barycentric weights used for the interpolation of the ith vertex element.

To generate the barycentric coordinates $\langle b_{i,A}, b_{i,B}, b_{i,C} \rangle$ needed for the interpolation of the ith vertex element, the framework randomly generating two pairs of barycentric coordinates; one pair of coordinates $\langle b_{root,A}, b_{root,B}, b_{root,C} \rangle$, determines the weights used in the interpolation of the first vertex elements, located at the root of the hair strand, and one pair of coordinates $\langle b_{tip,A}, b_{tip,B}, b_{tip,C} \rangle$, determines the weight used in the interpolation of the last vertex elements, located at the tip of the hair strand. The intermediate weights are then linearly interpolated with the following equations:

$$b_{i,A} = b_{root,A} + \frac{\left(b_{tip,A} - b_{root,A}\right) i}{n}$$

$$b_{i,B} = b_{root,B} + \frac{\left(b_{tip,B} - b_{root,B}\right) i}{n}$$

$$b_{i,C} = b_{root,C} + \frac{\left(b_{tip,C} - b_{root,C}\right) i}{n}$$

where n represents the total number of vertex elements.

As mentioned earlier in this section, the interpolated hairs are evenly spaced near their roots and clump, to the closest guide hair, near their tips. To get an even spacing of the interpolated hairs near the roots, all valid barycentric coordinates in the triangular surface should have an equal chance of being generated. The algorithm used in the framework is listed below:

- Generate two random values v_1 and v_2 in the range of $[0..1]$:

$$v_1 \longleftarrow \mathbf{random}(0,1)$$

$$v_2 \longleftarrow \mathbf{random}(0,1)$$

- If $v_1 + v_2 > 1$, set the largest of the two values to 1 minus the value:

$$\mathbf{IF}\ (v_1 > v_2)\ \{\, v_1 \longleftarrow 1 - v_1 \,\}$$

$$\mathbf{IF}\ (v_2 > v_1)\ \{\, v_2 \longleftarrow 1 - v_2 \,\}$$

- Set a third value v_3 to 1 minus the other two values:

$$v_3 \longleftarrow 1 - (v_1 + v_2)$$

- Assign the three values to the corresponding barycentric variable:

$$b_{root,A} \longleftarrow v_1$$

$$b_{root,B} \longleftarrow v_2$$

$$b_{root,C} \longleftarrow v_3$$

This algorithm results in randomly generated barycentric coordinates which sum up to the value $K = 1$. The barycentric coordinates stored v_1 and v_2 are randomly chosen in the interval $[0 \dots 1]$. Because $v_1 + v_2 + v_3 = q$, the following criteria must hold: $v_1 + v_2 < 1$. This criterion is met by resetting the larger of the two values to 1 minus that value, when $v_1 + v_2 > 1$. When two barycentric coordinates are known, the last is constrained by $K - (b_A + b_B)$, thus the last value becomes $v_3 = 1 - (v_2 + v_3)$

To get a clumping behaviour of the tip of the interpolated hairs with the closest guide hair, all the generated barycentric coordinates should be located within a specified distance of the guide hair closest to the interpolated hair. The closest guide hair is specified as the guide hair which has its root located closest to the root of the interpolated hair. Because the roots of the guide hairs are located at the corner points of the triangular surface, this guide hair can be identified as the guide hair belonging to largest generated barycentric variable in the root coordinates. The algorithm used in the framework is listed below:

- Generate two random values v_1 and v_2 in the range of $[0..clump]$:

$$v_1 \longleftarrow \mathbf{random}(0, clump)$$

$$v_2 \longleftarrow \mathbf{random}(0, clump)$$

- If $v_1 + v_2 > 1$, set the largest of the two values to 1 minus the value:

$$\mathbf{IF}\ (v_1 > v_2)\ \{\ v_1 \longleftarrow 1 - v_1\ \}\ \mathbf{ELSE}\ \{\ v_2 \longleftarrow 1 - v_2\ \}$$

- Set a third valued v_3 to 1 minus the other two values:

$$v_3 \longleftarrow 1 - (v_1 + v_2)$$

- Assign the three values to the corresponding barycentric variable:

$$b_{tip,A} \longleftarrow \left(b_{root,A} < b_{root,B}\right)\ ?\ v_1 : \left(\left(b_{root,A} < b_{root,C}\right)\ ?\ v_2 : v_3\right)$$

$$b_{tip,B} \longleftarrow \left(b_{root,B} < b_{root,C}\right)\ ?\ v_1 : \left(\left(b_{root,B} < b_{root,A}\right)\ ?\ v_2 : v_3\right)$$

$$b_{tip,C} \longleftarrow \left(b_{root,C} < b_{root,A}\right)\ ?\ v_1 : \left(\left(b_{root,C} < b_{root,B}\right)\ ?\ v_2 : v_3\right)$$

This algorithm results in the generation of barycentric coordinates which will clump to the guide hair that has its root located closest to the root of the interpolated hair. The variable $clump$ – where $0 \leq clump < 1$ – restricts the barycentric coordinates, stored in v_1 and v_2, from lying too close to the guide hairs which distance between their roots and the root of the interpolated hair is not minimal. The barycentric coordinate, stored in v_3, will always be assigned to the guide hair which has its root closest to the root of the interpolated hair strand. The other two barycentric coordinate stored in v_1 and v_2 respectively, should be assigned to the other two guide hairs. Low values for the variable $clump$ result in tighter clumping behaviour of the interpolated hair, while higher values for the variable $clump$ will result in a looser clumping behaviour of the interpolated hair. In this project $clump$ is set to $clump = 0.25$, which will results in a believable hair distribution.

Figure 30 contains two screenshots taken from within the hair framework. The left image shows the virtual characters head with guide hairs growing from the follicles of the dedicated wick, and the right image shows the same virtual characters head after the interpolation process; the interpolated hairs form a complete hair coupe.

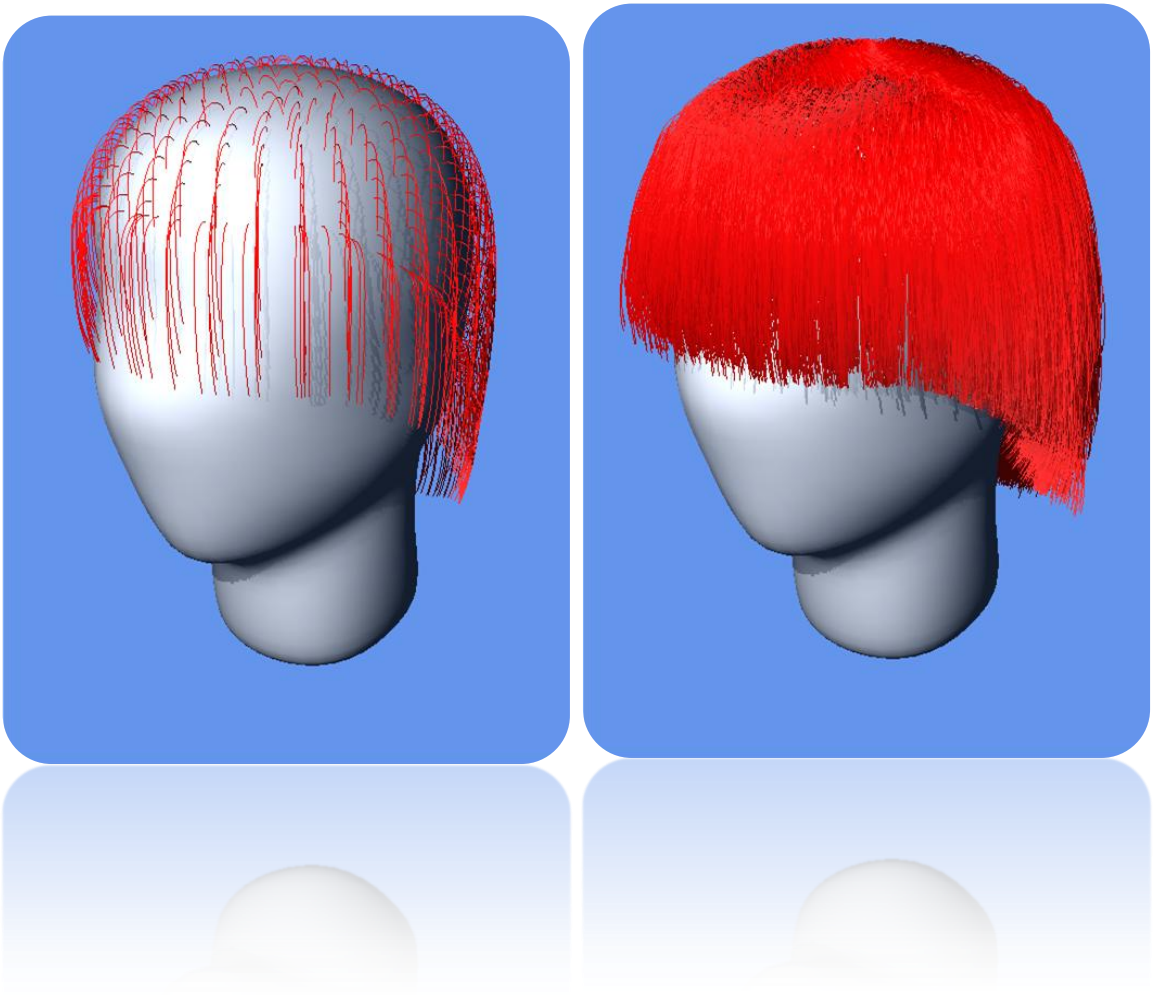

Figure 30. Hair Interpolation

4.1.4.4. Stray Hairs

An unexpected behaviour was discovered during one of the revisions of the interpolation algorithms described in previous sub-section. The intention of the algorithm, that generates the barycentric coordinates for the tip of the interpolated hair strand, was to generate two restricted values, v_1 and v_2, in the range of $[0 \dots clump]$, and determine the third value v_3, such that $v_1 + v_2 + v_3 = 1$. The "unrestricted" value v_3 need to be assigned to the barycentric variable belonging to the guide hair that has its root located closest to the root of the interpolated hair, and the other two values need to be assigned to the other two barycentric variables.

Value v_3 is, as expected, always assigned to the correct barycentric variable, but there exist situations where the value of v_1 is assigned to both remaining barycentric variables while the value of v_2 stays unused. There are six possible cases for the algorithm, which are listed in Table 1. From this table it becomes clear that in 50% of the cases v_1 is assigned to two barycentric variables, while v_2 stays unused.

Table 4. Barycentric Coordinate Assignment

Case	Assigned to $b_{tip,A}$	Assigned to $b_{tip,B}$	Assigned to $b_{tip,C}$
$b_{root,A} > b_{root,B} > b_{root,C}$	v_3	v_2	v_1
$b_{root,A} > b_{root,C} > b_{root,B}$	v_3	v_1	v_1
$b_{root,B} > b_{root,C} > b_{root,A}$	v_1	v_3	v_2
$b_{root,B} > b_{root,A} > b_{root,C}$	v_1	v_3	v_1
$b_{root,C} > b_{root,A} > b_{root,B}$	v_2	v_1	v_3
$b_{root,C} > b_{root,B} > b_{root,A}$	v_1	v_1	v_3

Work has been done to remove this unintentional behaviour from the algorithm by rewriting the conditional assignments so that – in all cases – both v_1 and v_2 are assigned to the barycentric variables. However, against expectations, this correction of the algorithm reduces the quality and believability of the rendered images. Decided is to revert the algorithm to its previous behaviour. In the remainder of the section I will explain the effect of this decision.

Figure 31 contains a screenshot taken from within the hair framework. The image shows the virtual characters head with the hair interpolated after the correction is applied. Interpolation artefacts are seen near the edges of the guide hairs. Also, the hair appears unrealistic sharp and flat – lacking the presence of stray hairs seen in the hair interpolation before the correction of the algorithm.

Figure 31. Revised Hair Interpolation

When the value of v_1 is assigned to two barycentric variables, the sum of the barycentric coordinates will not be $K = 1$ anymore. Because in the case $v_1 \neq v_2$ the following inequality holds:

$$v_1 + v_2 + v_3 \neq v_1 + v_1 + v_3$$

Whether the sum K of the barycentric coordinates is larger or smaller than 1 depends entirely on the relation between v_1 and v_2. In the case $v_1 > v_2$, the sum will be larger than 1, and in the case, $v_2 > v_1$, the sum will be smaller than 1. Because the values v_1 and v_2 are chosen at random in a domain of $[0..clump]$ both cases have a 50% change of happening, and as the maximum difference between v_1 and v_2 is $clump$, K will have a value in the range of $[1 - clump \ldots 1 + clump]$.

If $K \neq 1$, the positions of the tip interpolated vertex element will no longer be located inside the triangular area formed by the tip vertex elements of the guide hairs. When $K < 1$, the position of the vertex element will be smaller and will be located closer to the origin of the coordinate system, and when $K > 1$, the position of the vertex element will be larger and will be located further away from the origin of the coordinate system. The size of the tangent vector of the vertex element is also affected when $K \neq 1$, but will be normalized during the rendering stage. The origin of the coordinate system is located at the centre of the virtual character head.

By leaving the unintentional behaviour in the algorithm, the interpolation algorithm will produce the following results:

- In 50% of the cases the barycentric coordinates sum up to a value $K = 1$, and the tip of the interpolated hair will be positioned inside the triangular area formed by the tips of the three corresponding guide hairs.
- In 25% of the cases the barycentric coordinates sum up to a value $K < 1$, and the tip of the interpolated hair will have a position closer to the head than the triangular area formed by the tips of the three corresponding guide hairs.
- In 25% of the cases the barycentric coordinates sum up to a value $K > 1$, and the tip of the interpolated hair will have a position further away from the head than the triangular area formed by the tips of the three corresponding guide hairs.

Above described behaviour of the algorithm creates the stray hears visible in Figure 30. This behaviour effectively hides the interpolation artefacts and removes the unrealistic sharp and flat appearance of the hair coupe seen in Figure 31.

Once the entire hair coupe is interpolated, the position and tangent attributes of all the vertex elements – that make up the individual hair strands in the generated hair coupe – are stored into one single large Vertex Buffer Object (VBO). The framework places the VBO in the graphical memory and hints the GPU that the VBO is intend to only contain static data. This setup enables the GPU to efficiently retrieve the vertex information from cache, instead of requiring that the framework needs to re-send the entire vertex information for every single frame rendered.

The vertex elements in the VBOs will form line segments used by the hair vertex and fragment shaders to render the graphical representation of the individual hair strands.

4.1.5. Hair Illumination

The illumination of the hair is taken care of by the local illumination model introduced by (Kajiya, et al., 1989). The model of (Kajiya, et al., 1989) has two major advantages over other illumination models. The first advantage is that it is the single simplest shading algorithm to understand and to implement. And second, due its simplicity, it requires only few calculations and is thus also a very fast algorithm for the hair illumination. (Kajiya, et al., 1989) motivates that although the algorithm is not entirely physical correct, it can produce images of adequate quality.

4.1.5.1. Kajiya & Kay

As mentioned earlier in section ***Illumination***, located in chapter ***Related Work***, the illumination model of (Kajiya, et al., 1989) follows directly from the underlying cylindrical nature of a human hair strand. Figure 32Figure 11 shows an element of hair and the important vector quantities used in the model. The unit tangent vector ($\boldsymbol{t}$), represents the direction of the hair axis. The light vector ($\boldsymbol{l}$) points in the direction of the light source. The reflection vector ($\boldsymbol{r}$) points in the direction of reflected light and is the direction of maximum specular reflection. The eye vector ($\boldsymbol{e}$) points in the direction of the observer. The angles θ and ϕ are the angles from the tangent vector to the light vector and from the tangent vector to the eye vectors respectively.

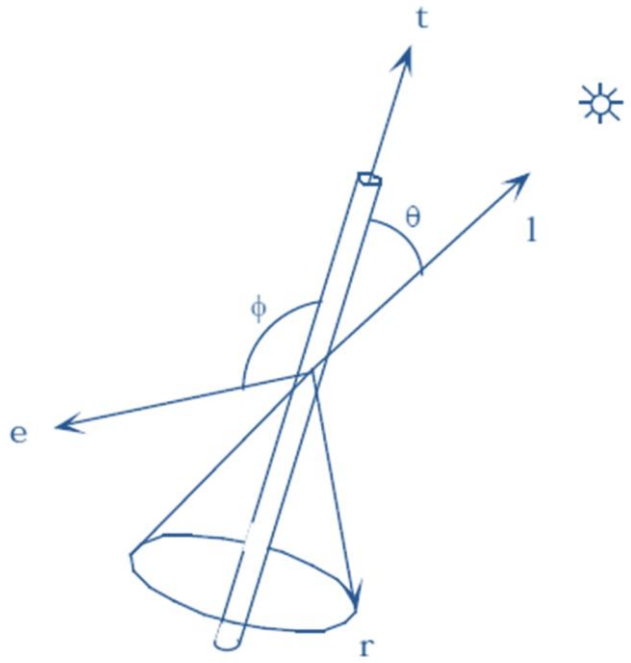

Figure 32. Kajiya & Kay Illumination Model

The diffuse component of the hair illumination model can be obtained by integrating the Lambert illumination model along the illuminated half of the hair cylinder; the back of the surface does not contribute to the calculation. The Lambert model calculates the intensity of reflected light by taking the dot product between the light vector and the normal vector and multiplying the results with the diffuse reflection coefficient, and is specified by the following equation:

$$I_{lambert} = K_d\ (\boldsymbol{l} \cdot \boldsymbol{n})$$

where K_d is the coefficient of diffuse reflectance. With the Lambert model, the intensity becomes *one* when the normal $\boldsymbol{n}$ and light vector $\boldsymbol{l}$ point in the same direction and the intensity becomes *zero*

when the normal $\boldsymbol{n}$ and light vector $\boldsymbol{l}$ are perpendicular to each other. This behaviour can also been seen in real life, where a surface appears to have a higher reflective intensity when the angle of incoming light strikes the surface at a larger angle.

To integrating the Lambert model along the illuminated half of the hair cylinder, a orthonormal base is formed by three vectors $\boldsymbol{t}$, $\boldsymbol{l}'$ and $\boldsymbol{b}$, where: vector $\boldsymbol{t}$ is the tangent along the cylinder, vector $\boldsymbol{l}'$ is the projection of the light vector $\boldsymbol{l}$ onto plane P containing all the normal vectors to the cylinder, and vector $\boldsymbol{b}$ is the orthonormal vector to both $\boldsymbol{t}$ and $\boldsymbol{l}'$. Vector $\boldsymbol{l}'$ can be calculated with the following equation:

$$\boldsymbol{l}' = \frac{\boldsymbol{l} - (\boldsymbol{t} \cdot \boldsymbol{l})\boldsymbol{t}}{|\boldsymbol{l} - (\boldsymbol{t} \cdot \boldsymbol{l})\boldsymbol{t}|}$$

and vector $\boldsymbol{b}$, orthogonal to $\boldsymbol{t}$ and $\boldsymbol{l}'$, and can be calculated with a cross product:

$$\boldsymbol{b} = \boldsymbol{l}' \times \boldsymbol{t}$$

Once the orthonormal base is specified, all the normals, located at the illuminated side of the hair cylinder, are calculated with the use of trigonometric functions. The normals are calculated with the following equation:

$$\boldsymbol{n}(x) = \boldsymbol{b} \cos x + \boldsymbol{l}' \sin x$$

where $x \in [0..\pi]$ specifies the location in radians along the illuminated semicircle; The location x is bound by the two shadow edges located at $x = 0$ and $x = \pi$. Thus to find the total amount of light per unit length, the integral of the Lambert cosine law must be taken, along the circumference of the illuminated half cylinder. The equation becomes:

$$I_{diffuse} = K_d \int_0^{\pi} [\boldsymbol{l} \cdot \boldsymbol{n}(x)]\, dx$$

When expanding $\boldsymbol{n}(x)$, the equation becomes:

$$I_{diffuse} = K_d \int_0^{\pi} [\boldsymbol{l} \cdot (\boldsymbol{b} \cos x + \boldsymbol{l}' \sin x)]\, dx$$

because $\int_0^{\pi} [\cos x] = 0$:

$$I_{diffuse} = K_d \int_0^{\pi} [\boldsymbol{l} \cdot \boldsymbol{l}' \sin x]\, dx$$

and because $\int_0^{\pi} [\sin x] = 1$:

$$I_{diffuse} = K_d\, (\boldsymbol{l} \cdot \boldsymbol{l}')$$

When expanding $\boldsymbol{l}'$, the equation becomes:

$$I_{diffuse} = K_d \left(\boldsymbol{l} \cdot \frac{\boldsymbol{l} - (\boldsymbol{t} \cdot \boldsymbol{l})\boldsymbol{t}}{|\boldsymbol{l} - (\boldsymbol{t} \cdot \boldsymbol{l})\boldsymbol{t}|} \right)$$

which in its turn becomes:

$$I_{diffuse} = K_d \left(\frac{1 - (\boldsymbol{t} \cdot \boldsymbol{l})^2}{\sqrt{1 - (\boldsymbol{t} \cdot \boldsymbol{l})^2}} \right)$$

which in its turn becomes:

$$I_{diffuse} = K_d \sin(\boldsymbol{t}, \boldsymbol{l})$$

which yield a simple function depended only on the angle θ. The diffuse term can be calculated with the following equation:

$$I_{diffuse} = K_d \sin\theta$$

The diffuse component becomes proportional to the sine between the light and tangent vectors. Thus if the tangent of the hair is pointing straight at the light, the hair is dark. This can also be observed in real hair. The function does not take into account self-shadowing, that is, the fact that half of the hair strand is in shadow. Therefore, when the hair would be an opaque cylinder, the diffuse component would vary with what visible fraction is in shadow, which is dependent on the direction of the eye vector $\boldsymbol{e}$. However, human hairs are actually quite translucent so that the area in shadow transmits diffusely an amount of light comparable to the amount reflected diffusely on the non-shadow side of the hair. As a result, this simple formula makes an approximation that works very well in practice.

The specular component of the hair illumination model can be obtained by calculating the highlights on the hair. Any light striking the hair is specularly reflected at a mirror angle along the tangent. At any section along the length of the hair, there are surface normals pointing in all directions perpendicular to the tangent vector $\boldsymbol{t}$. Therefore, the 180 degree semicircular surface of the hair, which is not in shadow, will directly reflect light in a cone-shaped 360 degree radial pattern formed by rotating the reflection vector $\boldsymbol{r}$ around the hair axis. Since the normals on the cylinders point in all directions perpendicular to the tangent, the reflected light should be independent of the component of the eye vector which is parallel to plane P. Thus the reflected light forms the cone whose angle at the apex is equal to the angle of incidence. The actual highlight intensity is given as:

$$I_{specular} = K_s \cos^p(\boldsymbol{e}, \boldsymbol{r})$$

where K_s is the coefficient of specular reflectance, vector $\boldsymbol{e}$ is the vector pointing to the eye, vector $\boldsymbol{r}$ is the specular reflection vector contained in the cone closest to the eye vector, and scalar p is the exponent specifying the sharpness of the highlight. The highlight is thus a maximum when the eye vector is contained in the reflected cone and falls off with an exponential dependence when the vectors diverge.

To calculate this model the only quantities entering into the calculation are angle θ', representing the angle between the tangent vector $\boldsymbol{t}$ and the reflection vector $\boldsymbol{r}$, and angle ϕ, representing the angle between the tangent vector $\boldsymbol{t}$ and the eye vector $\boldsymbol{e}$. The intensity is given by:

$$I_{specular} = K_s \cos^p(\theta' - \phi)$$

Because the angle between the tangent vector and reflection vector is similar to the angle between the tangent vector and the light vector:

$$\theta' = \theta$$

The diffuse equation can be rewritten as:

$$I_{specular} = K_s \cos^p(\theta - \phi)$$

which yield a simple function depended only on the angle between the tangent and light vector and on the angle between the tangent and eye vector. The specular term can be calculated with the following equation:

$$I_{specular} = K_s(\cos\theta\cos\phi + \sin\theta\sin\phi)^p$$

Once the diffuse and specular components are calculated, they can be combined together with the ambient component to form the final hair illumination equation:

$$I_{KajiyaKay} = L_a K_a + \sum_{lights} \left(L_i(K_d \sin\theta + K_s(\cos\theta\cos\phi + \sin\theta\sin\phi)^p)\right)$$

Where L_i is the light intensity, K_a is the ambient reflectance coefficient, and L_a is the ambient light intensity which is present in the environment.

The hair vertex and fragment shaders, used for the illumination calculations of the hair coupe, can be found in section ***Hair Shader***, located in ***Appendix A***. In the section below: the **Red** coloured numbers are used to point to the line numbers of the hair vertex shader program, while the **Green** coloured numbers are used to point to the line numbers of the hair fragment shader program.

The vectors $\boldsymbol{t}$, $\boldsymbol{l}$ and $\boldsymbol{e}$ are calculated in the hair vertex shader. Tangent vector $\boldsymbol{t}$ and position $\boldsymbol{p}$ are retrieved at line **17** and **18** by transforming them from world space coordinates to camera space coordinates. The light vector $\boldsymbol{l}$ – pointing from position $\boldsymbol{p}$ to the light source – is retrieved at line **19** by transforming the light position from world space coordinates to camera space coordinates and then subtracting the position $\boldsymbol{p}$.

Because in eye space the camera is placed at the centre of the coordinate system, eye vector $\boldsymbol{e}$ – pointing from position $\boldsymbol{p}$ to the camera – is retrieved at line **21** by mirroring the value of $\boldsymbol{p}$. After the vectors $\boldsymbol{t}$, $\boldsymbol{l}$ and $\boldsymbol{e}$ are retrieved, they will be passed on to the fragment shader, where they will be used in the illumination calculations.

The hair illumination equation of (Kajiya, et al., 1989) is evaluated in the hair fragment shader. Tangent vector $\boldsymbol{t}$, light vector $\boldsymbol{l}$ and eye vector $\boldsymbol{e}$ are normalized at line **26**, **27** and **28** to ensure that the vectors are of unit length. $\cos\theta$, $\cos\theta$, $\sin\theta$ and $\sin\phi$ are calculated at line **31**, **32**, **34** and **35**; these values can be used in the calculation of the diffuse and specular component. The ambient reflective intensity is set to a constant value at line **38**, and the diffuse and specular reflective intensities are calculated at line **39** and **40**.

Once all the intensities are evaluated, The final hair colour can be calculated by multiplying the ambient, diffuse and specular intensities with the corresponding colours and summing the results together. The colour is calculated at line **71**, **72**, **73** and **74** of the fragment shader. The calculation of the shadow terms, which lower the intensity of the diffuse and specular terms, will be explained later in section ***Hair Shadowing***.

Figure 33 contains three screenshots taken from within the hair framework. The left image shows a hair coupe illuminated only by the diffuse intensity, the middle image shows a hair coupe illuminated only be the specular intensity, and the right image shows a hair coupe illuminated by the diffuse intensity and the specular intensities combined.

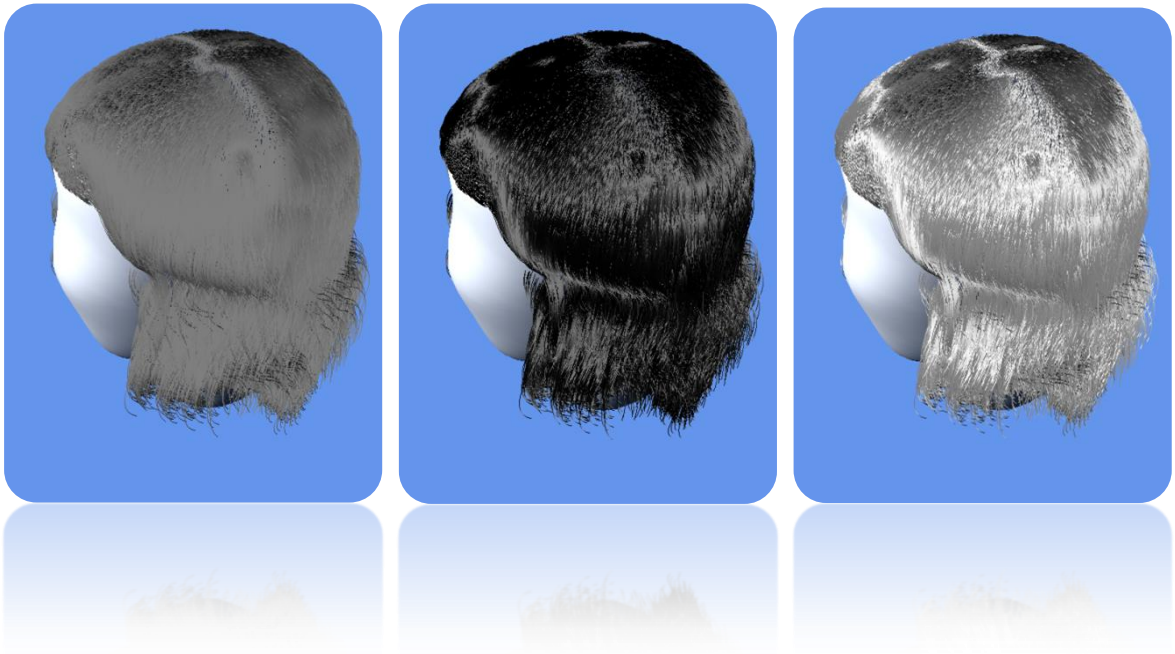

Figure 33. Diffuse and Specular Terms

The hair coupe is not the only object in the scene that needs to be illuminated during the rendering. The head vertex and fragment shaders, used for the illumination calculations of the virtual character head, can be found in section ***Head Shader***, located in ***Appendix A***. In the section below: the **Orange** coloured numbers are used to point to the line numbers of the head vertex shader program, while the **Purple** coloured numbers are used to point to the line numbers of the head fragment shader program.

The vertex and fragment shaders for the head are very similar in design as the vertex and fragment shaders for the hair, and should require little explanation. The only difference in the head vertex shader is that a normal vector $\boldsymbol{n}$ is calculated instead of the tangent vector $\boldsymbol{t}$. The normal vector $\boldsymbol{n}$ is retrieved at line **17** by transforming it from world space coordinates to camera space coordinates.

In the head fragment shader, the Lambert model does not have to be integrated around a semicircle, but can instead use the normalized normal vector from line **23** directly for calculating the diffuse intensity at line **29**. The intensity is calculated with the following equation:

$$I_{diffuse} = K_d\,(\boldsymbol{l} \cdot \boldsymbol{n})$$

Once the intensity is evaluated, the final hair colour can be calculated by multiplying the ambient and diffuse intensities with the corresponding colours and summing the results together. The colour is calculated at line **58**, **59** and **60** of the fragment shader. The calculation of the shadow terms, which lowers the intensity of the diffuse term, will be explained later in section ***Hair Shadowing***.

The left screenshot in Figure 27 already showed the results which can be acquired when rendering with the head shaders. Surfaces oriented perpendicular to the incoming light rays have a higher reflective intensity than the surfaces oriented parallel to the incoming light rays.

4.1.6. Hair Shadowing

Shadows are added to the hair by implementing the deep opacity maps developed by (Yuksel, et al., 2008). The reason for choosing to implement the shadow technique of (Yuksel, et al., 2008), is that (Yuksel, et al., 2008) have proven, through a comparison of several shadow techniques, that their technique is superior over others in both the performance of the algorithm and the quality of the shadows generated.

4.1.6.1. Deep Opacity Maps

As mentioned earlier in section ***Illumination***, located in chapter ***Related Work***, deep opacity maps use an algorithm that needs two passes to prepare the shadow information, and needs an additional pass to render the final images with shadows.

The first step prepares the separation between the opacity layers. A depth map (shadow map) is rendered as seen from the light source, which gives a per pixel depth value Z_0 at which the hair geometry begins. Starting from this depth value, the hair volume is divided into K layers such that each layer lies further from the light source than the previous layer. Because the algorithm uses a shadow map to determine where the K layers are positioned, the layers will take the shape of the hair structure.

The second step renders the opacity map using the depth map computed in the previous step. This requires that the hair needs to be rendered only once. As each hair is rendered, the Z_0 value from the depth map is read to find the depth values of each layer on the fly. The opacity contribution of a hair fragment is added to the corresponding layer in which the hair fragment falls and all layers behind it. The total opacity of a layer at a pixel is the sum of all the contributing fragments.

The opacity map can be represented by associating each colour channel with a different layer, and accumulate the opacities by using additive blending on the graphics hardware. Thus, by using a single colour value with four channels, four opacity layers can be stored. Multiple texture lookups can be used when one wants to use more opacity layers during the final rendering.

One disadvantage of using a small number of layers with deep opacity maps is that it can be more difficult to ensure that all points in the hair volume are assigned to a layer. In particular, points beyond the end of the last layer do not correspond to any layer. Mapping these points onto the last layer, will usually give good results. Unlike opacity shadow maps, deep opacity maps hide the interpolation of the opacity values within the hair volume, thus hiding possible layering artefacts.

Figure 34 shows an illustration of the deep opacity shadow map. The hair volume is sliced into different layers by using a previous calculated shadow map and the densities are stored together with the shadow map into the different colour channels of the deep opacity map. The deep opacity map is used during the final rendering to calculate how much light penetrates to the corresponding layer and thus how much of the hair is in shadow.

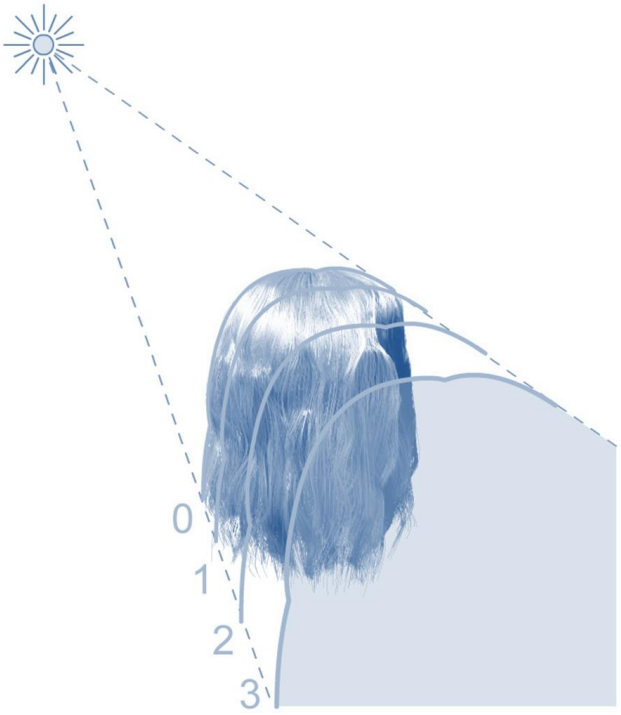

Figure 34. Deep Opacity Maps

The opacity vertex and fragment shaders, used for creating the opacity map, can be found in section ***Opacity Shader***, located in ***Appendix A***. In the section below: the **Red** coloured numbers are used to point to the line numbers of the opacity vertex shader program, the **Green** coloured numbers are used to point to the line numbers of the opacity fragment shader program.

After the depth map is rendered as seen from the position of the light source, the depth values stored in the depth texture need to be projected onto the object geometry. In the opacity vertex shader, the texture coordinates for the projection are generated at line **14** by multiplying the vertex position with a texture matrix. This texture matrix contains the projectors projection matrix which will transform the vertex position into the clip space of the projector. The projector is located at the position of the light source and is pointing to the centre of the head. The creation of the texture matrix will be explained later in section ***Texture Coordinates***.

In the opacity fragment shader the opacity values are calculated. The shadow depth Z_0 is retrieved at line **14** by doing a projective texture lookup on the shadow map with the texture coordinates calculated in the opacity vertex shader.

The next step is to transform the user specified layer sizes and the generated texture coordinates into homogeneous coordinates by dividing these vectors at line **16** and **17** by the Q component of the texture coordinate. Once the layer sizes are homogeneous they can be used in line **20** to line **24** to create the separations between the opacity layers; starting from the depth value Z_0, the hair volume is divided into 4 layers, such that each layer Z_i lies further from the light source than the previous layer.

In line **26** to line **47**, as each hair is rendered, the depth values Z_i are tested against the depth component of the homogenous texture coordinate and the opacity contribution is added to the

corresponding layer in which the hair fragment falls and all layers behind it. Points beyond the end of the last layer do not correspond to any layer, but will be added to the last layer instead.

The opacity map is represented by associating each colour channel with a different layer, and opacities are accumulated by using additive blending on the graphics hardware. The opacity maps can then be used in the hair shaders to calculate the shadow term of a fragment.

The hair vertex and fragment shaders – that retrieve the shadow term of the hair coupe from the opacity map, and use it in the equations of the final hair colour – can be found in section ***Hair Shader***, located in ***Appendix A***. In the section below: the Orange coloured numbers are used to point to the line numbers of the hair vertex shader program, and the Purple coloured numbers are used to point to the line numbers of the hair fragment shader program.

In the hair vertex shader, the texture coordinates for the projection are generated at line **14** by multiplying the vertex position with a texture matrix. This texture matrix contains the projectors projection matrix which will transform the vertex position into the clip space of the projector. The projector is located at the position of the light source and is pointing to the centre of the head. The creation of the texture matrix will be explained later in section ***Texture Coordinates***.

In the hair fragment shader the shadow term needs to be calculated. Similar in process as the opacity fragment shader, the shadow depth Z_0 is retrieved at line **22** by doing a projective texture lookup on the shadow map with the texture coordinates calculated in the hair vertex shader. The opacity values are retrieved at line **23** by doing a projective texture lookup on the opacity map.

The next step is to transform the user specified layer sizes and the generated texture coordinates into homogeneous coordinates by dividing these vectors at line **43** and **44** by the Q component of the texture coordinate. Once the layer sizes are homogeneous they can be used in line **46** to line **50** to create the separations between the opacity layers; starting from the depth value Z_0, the hair volume is divided into 4 layers, such that each layer Z_i lies further from the light source than the previous layer.

In line **56** to line **66**, the depth values Z_i are tested against the depth component of the homogenous texture coordinate, and the shadow value is found by interpolating between the two corresponding opacity values of the layer in which the hair fragment falls. Points beyond the end of the last layer will use the opacity value of the last layer as value.

Because a high opacity should result in a large decrease of intensity, the shadow is scaled and biased in line **68** by mapping the initial shadow in the range $[0 \dots 1]$ to the range $[1 \dots 0{,}25]$. At this point, the shadow term can be used to lower the intensities of the diffuse and specular term by multiplication.

As mentioned earlier, the vertex and fragment shaders for the head – found in section ***Head Shader***, located in ***Appendix A*** – are very similar in design as the vertex and fragment shaders for the hair. This applies also to the shadow calculations. The calculations needed for the shadow term are identical for both the hair shader and the head shader, apart from the differences in line numbers where the equations are located.

Figure 35 contains two screenshots taken from within the hair framework. The left image shows a hair coupe illuminated without the use of a shadow term, and the right image shows a hair coupe illuminated with the use of a shadow term. The hair coupe rendered with proper shadows appears to have more depth and looks much more realistic.

Figure 35. Hair Self-Shadowing

4.1.6.2. Texture Coordinates

For projective texturing the shadow and deep opacity maps, per-vertex texture coordinates cannot be explicitly specified by the application; the texture coordinates need to be calculated in the vertex program from the object-space per-vertex positions automatically.

Figure 36 shows the sequence of transformations that need to be implemented in the vertex program to be able to perform projective texturing. This sequence of transformations is the same sequence that would be used when the camera would be located at the light source position, but with one extra matrix concatenated. This last matrix scales and biases the resulting coordinates into the range of $[0 \dots 1]$ which is needed to access textures. For efficiency, it is best to concatenate the matrices into a single matrix. This concatenated matrix is called the texture matrix.

$$\begin{bmatrix} s \\ t \\ r \\ q \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & & & \frac{1}{2} \\ & \frac{1}{2} & & \frac{1}{2} \\ & & \frac{1}{2} & \frac{1}{2} \\ & & & 1 \end{bmatrix} \begin{bmatrix} Light \\ Frustum \\ (projection) \\ Matrix \end{bmatrix} \begin{bmatrix} Light \\ View \\ (look\ at) \\ Matrix \end{bmatrix} \begin{bmatrix} \\ Modeling \\ Matrix \\ \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ w_0 \end{bmatrix}$$

Figure 36. Projective Texturing

When vertex positions are multiplied with the texture matrix, they will undergo several matrix transformations. The vertex coordinates $\langle x, y, z, w \rangle$ start in object space and transform into projective texture coordinates $\langle s, t, r, q \rangle$ by undergoing the following matrix multiplications:

- **Multiplying by the Modelling Matrix**. This transforms the object space vertex coordinates to world space vertex coordinates. This modelling matrix needs to be applied regardless of whether or not one is using projective texturing.
- **Multiplying by the Light's View Matrix**. This rotates and translates the world space vertex coordinates so that they are in the light's frame of reference.
- **Multiplying by the Light's Projection Matrix**. This defines the light's frustum, including its field of view and aspect ratio.
- **Scaling and Biasing the Results**. Following previous steps, the transformed vertex values range from -1 to $+1$. However, textures are indexed from 0 to 1, so the results have to be scaled and biased to this range. This is done by multiplying the x, y, and z components of the results by the value $\frac{1}{2}$ and then adding the value $\frac{1}{2}$.

The matrices can be acquired easily. The modelling matrix is already known, the light's viewing and projection matrices are the matrices used when rendering the scene from the light's point of view, and the scale and bias matrix is simply a collection of constants.

4.2.Results

By combining above techniques, the project has resulted in a working hair simulation & rendering framework which is able to display physical and visual believable human hair. A user of the framework is able to specify the hair and environment properties, and the framework will compute and generate the visual representation of the hair coupe.

In the sections below I will cover the user specified input for the framework, the workflow within the framework, and the output of the framework.

4.2.1. Input

A user of the framework should be able to specify the hair and environment properties which will influence the shape and appearance of the hair coupe rendered. These values can be specified by setting the parameters located in a set of four text documents. These four text documents are divided into two categories: one document contains the setup of the framework, and the other three documents contain information used in the framework.

The main text document – called *Setup* – is the setup document and should carry the filename `setup.txt`, and should be located in the root directory of the framework. This ensures that the framework can find the text document and read its content. The only purpose of the *Setup* document is to specify the file paths where the remaining input elements of the framework can be found; this enables the framework to load these elements.

The other three text documents are information documents containing the parameters needed for the equations used for hair mechanics, hair collision detection, and hair visualization; these documents are called *Hairstyle Info*, *Collision Info*, and *Visuality Info* respectively.

First, the *Setup* file will be described. An example of this file can be found in section ***Input Files***, located in ***Appendix B***. The list below contains all the property fields present in *Setup* and gives a description of the input parameters.

Setup:

- **Load**. Boolean value indicates whether the hair mesh should be loaded or generated. When load contains the value true, the hair mesh is loaded from a serialized hair object, otherwise the hair mesh is generated by the framework.
- **Save**. Boolean value indicates whether the hair mesh should be saved. When save contains the value true, the hair mesh will be saved into a serialized hair object.
- **LoadPath**. String specifying a path to the location where the serialized hair object can be loaded. This parameter will be ignored when **Load** contains the value false. The string should be enclosed by double quotation marks when the path contains spaces.
- **SavePath**. String specifying a path to the location where the serialized hair object should be saved. This parameter will be ignored when **Save** contains the value false. The string should be enclosed by double quotation marks when the path contains spaces.
- **HairstyleInfo**. String specifying a path to the location where the *Hairstyle Info* file can be found. This parameter will be ignored when **Load** contains the value true. The string should be enclosed by double quotation marks when the path contains spaces.
- **CollisionInfo**. String specifying a path to the location where the *Collision Info* file can be found. This parameter will be ignored when **Load** contains the value true. The string should be enclosed by double quotation marks when the path contains spaces.
- **VisualityInfo**. String specifying a path to the location where the *Visuality Info* file can be found. This parameter is also required when **Load** contains the value true. The string should be enclosed by double quotation marks when the path contains spaces.
- **WickModel**. String specifying a path to the location where the head object file can be found. This parameter will be ignored when **Load** contains the value true. The string should be enclosed by double quotation marks when the path contains spaces. The path should point to an OBJ-file.

- **HeadVert**. String specifying a path to the location where the head vertex shader can be found. The string should be enclosed by double quotation marks when the path contains spaces. The path should point to a GLSL-file.

- **HeadFrag**. String specifying a path to the location where the head fragment shader can be found. The string should be enclosed by double quotation marks when the path contains spaces. The path should point to a GLSL-file.

- **HairModel**. String specifying a path to the location where the wick object file can be found. The string should be enclosed by double quotation marks when the path contains spaces. The path should point to an OBJ-file.

- **HairVert**. String specifying a path to the location where the hair vertex shader can be found. The string should be enclosed by double quotation marks when the path contains spaces. The path should point to a GLSL-file.

- **HairFrag**. String specifying a path to the location where the hair fragment shader can be found. The string should be enclosed by double quotation marks when the path contains spaces. The path should point to a GLSL-file.

The *Setup* file specifies the path where the *Hairstyle Info* file can be found. The *Hairstyle Info* file specifies the mechanical parameters of the hair strand. An example of this file can be found in section ***Input Files***, located in ***Appendix B***. The list below contains all the parameter fields present in *Hairstyle Info*, gives a description of its behaviour, and indicates where the value is used.

***Hairstyle Info*:**

- **VisualityStep**. Float value specifying the distance Δs – along the curvilinear abscissa of the Super-Helix – between two adjacent vertex elements. A small value will result in a smooth hair strand, but will also increase memory usage on the GPU. The explanation of this value can be found in section ***Hair Layout & Growth***, located in ***Design***.

- **CollisionStep**. Float value specifying the distance Δs – along the curvilinear abscissa of the Super-Helix – between two adjacent collision samples. A small value will result in precise collision detection, but will also increase computation time. The explanation of this value can be found in section ***Hair Collision***, located in ***Design***.

- **MaxIter**. Integer value specifying the maximum number of iterations followed before the Rosenbrock method finishes. A high value will result in precise energy minimization, but will also increase computation time. The Rosenbrock method can be found in section ***Energy Minimization***, located in ***Design***.

- **Epsilon**. Float value specifying the minimum amount of convergence needed before the Rosenbrock Method finishes. A small value will result in precise energy minimization, but will also increase computation time. The Rosenbrock method can be found in section ***Energy Minimization***, located in ***Design***.

- **HairLength**. Float vector specifying the lengths ℓ_Q of the Super-Helix elements. The number of values in the vector is implicitly specifying the number of elements of the Super-Helix. The Super-Helix model can be found in section ***Cosserat Rods***, located in ***Design***.

- **HairDensity**. Float vector specifying the hair density of the hair coupe, used during the hair interpolation process. A high value will result in a full hair coup, but will also increase memory usage on the GPU. The hair interpolation process can be found in section ***Hair Layout & Growth***, located in ***Design***.

- **HairRadius1**. Float value specifying the major axis a of the elliptic cross section of the hair strand, required to compute the equation for the moment of inertia of an ellipse, used in the internal elastic energy calculations. The energy calculations can be found in section ***Cosserat Rods***, located in ***Design***.

- **HairRadius2**. Float value specifying the minor axis b of the elliptic cross section of the hair strand, required to compute the equation for the moment of inertia of an ellipse, used in the internal elastic energy calculations. The energy calculations can be found in section ***Cosserat Rods***, located in ***Design***.

- **HelixRadius**. Float specifying the helix radius r_h, required for the calculation of the intrinsic torsion and curvatures of the Super-Helix. The natural torsion and curvature computations can be found in section ***Cosserat Rods***, located in ***Design***.

- **HelixStep**. Float specifying the helix step size Δ_h, required for the calculation of the intrinsic torsion and curvatures of the Super-Helix. The natural torsion and curvature computations can be found in section ***Cosserat Rods***, located in ***Design***.

- **Poisson**. Float specifying the Poisson's ratio σ, required for the calculation of the internal elastic energy. The energy calculations can be found in section ***Cosserat Rods***, located in ***Design***.

- **Young**. Float specifying the Young's modulus E, required for the calculation of the internal elastic energy. The energy calculations can be found in section ***Cosserat Rods***, located in ***Design***.

- **Density**. Float specifying the hair density ρ, required for the calculation of the potential gravitational energy. The energy calculations can be found in section ***Cosserat Rods***, located in ***Design***.

- **Gravity**. Float specifying the gravity field g, required for the calculation of the potential gravitational energy. The energy calculations can be found in section ***Cosserat Rods***, located in ***Design***.

The *Setup* file specifies the path where the *Collision Info* file can be found. The *CollisionInfo* file specifies the collision geometry in the rendered scene. An example of this file can be found in section ***Input Files***, located in ***Appendix B***. The list below contains all the parameter fields present in *Collision Info*, gives a description of its behaviour, and indicates where the value is used.

Collision Info:

- **(**Zero or more**) Sphere**. Six Floats specifying the attributes of the corresponding collision sphere; the first Float specifies the volume, the next Float specifies the penalty, the next three Floats specify the centre of the geometry, and the last Float specifies the radius of the geometry. A description of the sphere collision geometry, and its attributes, can be found in section ***Hair Collision***, located in ***Design***.

- **(**Zero or more**) Ellipsoid**. Seventeen Floats specifying the attributes of the corresponding collision ellipsoid; the first Float specifies the volume, the next Float specifies the penalty, the next three Floats specify the centre of the geometry, the next three floats specify the Dimension of the geometry, and the last nine Floats specify the three vectors aligned to the main axis of the collision geometry. A description of the ellipsoid collision geometry, and its attributes, can be found in section ***Hair Collision***, located in ***Design***.

- **(**Zero or more**) Cuboid**. Seventeen Floats specifying the attributes of the corresponding collision cuboid; the first Float specifies the volume, the next Float specifies the penalty, the next three Floats specify the centre of the geometry, the next three floats specify the Dimension of the geometry, and the last nine Floats specify the three vectors aligned to the main axis of the collision geometry. A description of the cuboid collision geometry, and its attributes, can be found in section ***Hair Collision***, located in ***Design***.

The *Setup* file specifies the path where the *Visuality Info* file can be found. The *Visuality Info* file specifies the colours of the ambient, diffuse and specular illumination components of the hair shader. An example of this file can be found in section ***Input Files***, located in ***Appendix B***. The list below contains all the parameter fields present in *Visuality Info*, gives a description of its behaviour, and indicates where the value is used.

Visuality Info:

- **AmbientColor**. Three Floats specifying the red, green, and blue colour channels of the ambient reflectance coefficient K_a used in the Kajiya & Kay illumination model. The hair shader implementation can be found section ***Hair Illumination***, located in ***Design***.

- **DiffuseColor**. Three Floats specifying the red, green, and blue colour channels of the diffuse reflectance coefficient K_d used in the Kajiya & Kay illumination model. The hair shader implementation can be found section ***Hair Illumination***, located in ***Design***.

- **SpecularColor**. Three Floats specifying the red, green and blue colour channels of the specular reflectance coefficient K_s used in the Kajiya & Kay illumination model. The hair shader implementation can be found section ***Hair Illumination***, located in ***Design***.

- **Glossiness**. Integer value specifying the power p of the specular equation, used in the Kajiya & Kay illumination model. The hair shader implementation can be found section ***Hair Illumination***, located in ***Design***.

Specifying above properties allows a wide variety of hair shapes and hair appearances to be simulated and rendered by the framework.

4.2.2. Work Flow

Once a user of the framework has specified all the required parameters and has provided the elements which are referenced from within the *Setup* document, the framework will use the specified parameters and elements to simulate and render the virtual hair coupe. The framework will go through several processing steps to generate the final rendered image.

Figure 37 shows an illustration of the workflow within the hair simulation & rendering framework. In this illustration the cylinders represent information loaded from data files, and the boxes represents the states present within the framework. A comprehensive description of the processing steps and the data flow within the framework will be given below.

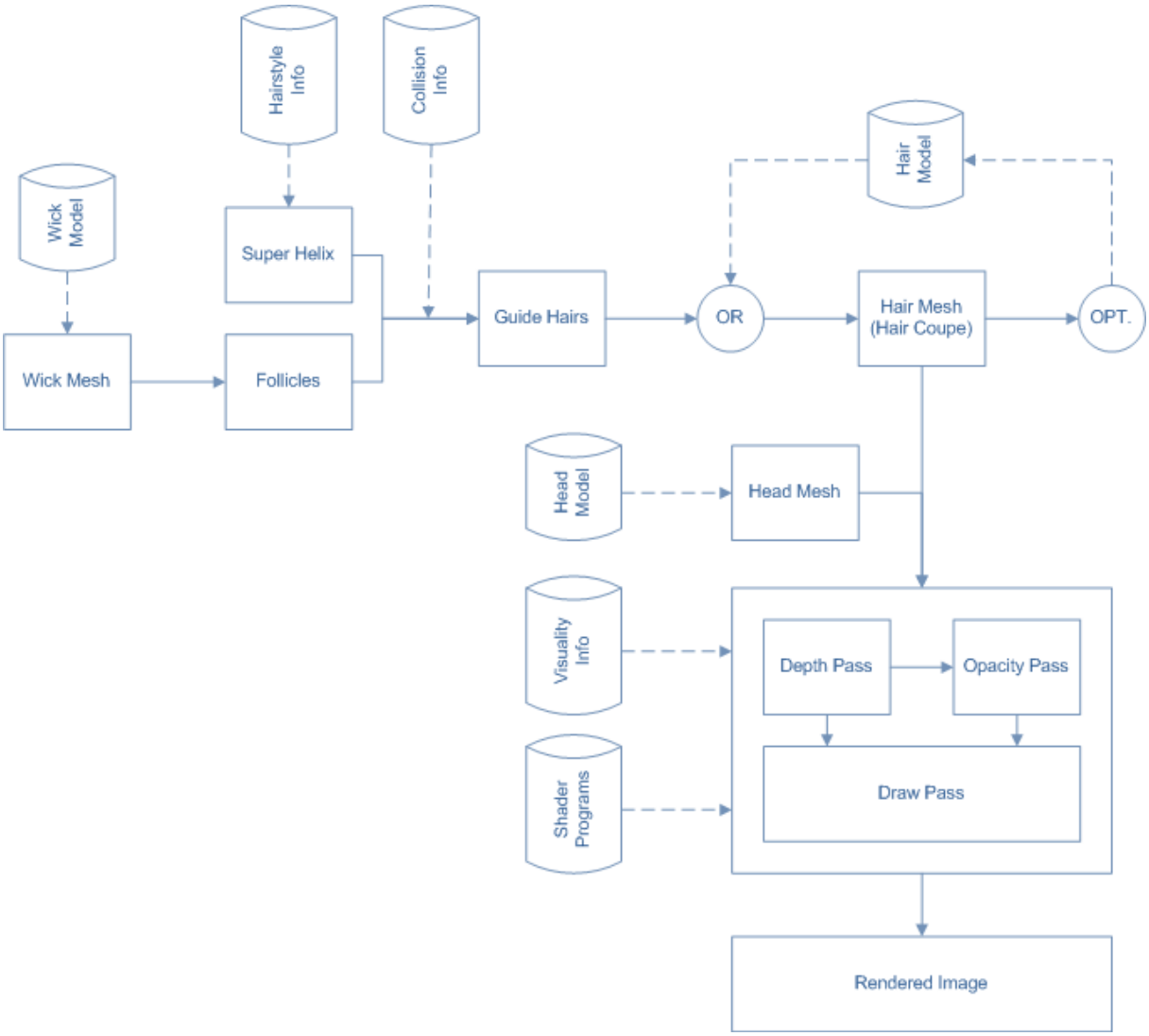

Figure 37. Workflow within Framework

The first step in the framework is to load the *wick model*. Every vertex element within the triangular *wick mesh* represents a hair *follicle*. The position and normal vector stored at every vertex element determines the location and the orthogonal base from which a *guide hair* should be grown. To allow the growth of the *guide hairs*, the mechanical parameters of the *Super-Helix* energy equation and the *Super-Helix* reconstruction equations need to be specified. This will be done by reading the mechanical parameters from the *Hairstyle Info* document and filling them into the equations.

Once the *Super-Helix* energy equations and the position and orthogonal base of the hair *follicles* are known, a minimization algorithm is used to find the unique configuration for which the *Super-Helix* internal energy becomes minimal; the collision penalties caused by the collision geometry listed in the *Collision Info* document are also considered in this energy minimization process. The unique configuration – for which the internal energy of the *Super-Helix* becomes minimal – will be used in the *Super-Helix* reconstruction process to construct the vertex representation of the corresponding *guide hairs*; the vertex elements of the guide *hair* will contain the position and tangent information of the *guide hair* taken at fixed intervals along the curvilinear abscissa of the hair strand.

The *guide hairs* alone will form a very sparse hair coupe. Additional hair strands need to be interpolated to fill up the remaining surface areas of the *wick mesh*. For every triangular face in the *wick mesh*, the three *guide hairs* growing form the corner vertices of the triangular face are interpolated across the surface. The interpolation is done with the help of randomly generated barycentric coordinates which act as weights in the interpolation calculations. The vertex attributes of the three *guide hairs* are interpolated, resulting in the creation of the vertex attributes of the newly interpolated hair. The interpolated hair will contain the same amount of vertices as the corresponding *guide hairs*. At this point, the vertex elements of the interpolated hair are stored into a vertex buffer object and will form the *hair mesh*. The number of interpolated hair strands that a surface will contain depends on the surface area and the hair density.

A user can specify whether he wants to save the generated *hair mesh* to disk, allowing him to load the *hair mesh* quickly at a later point in time, skipping the process of having to regenerate the same *hair mesh* over again.

Once the *hair mesh* is generated or loaded from disk, the framework will load the *head model*. The *hair mesh* and *head mesh* will form the objects that need to be rendered during the rendering stage of the framework:

- **The first** pass in the rendering stage uses no shaders, and renders the depth buffer to texture. This results in a depth texture containing the depth values of the head and wick mesh as seen from the light source position. This depth map will be used during the second and third pass.

- **The second** pass in the rendering stage uses the opacity shader, and renders the colour output to texture. It uses the depth values stored in the depth texture to divide the hair and head volumes into layers such that each layer lies further from the light source than the previous layer. By using additive blending on the graphics hardware, the accumulated opacity in each layer can be calculated. The opacity of each layer will be stored in the corresponding colour channel of the opacity map. This opacity map will be used during the third pass.

- **The third**, and final, pass in the rendering stage uses the Lambert model for rendering the head mesh, and uses the Kajiya & Kay model for rendering the hair mesh; the *Visuality Info* documents specifies the colouring of the hair coupe. The depth and opacity maps are used to retrieve the opacity value needed to determine the amount of shadow a fragment receives.

The colours calculated in the third pass will be written to the frame buffer, and the generated hair coupe will be displayed on the display.

4.2.3. Output

Once the hair coupe is generated, the framework will output the generated hair coupes to a display. Some example hairstyles generated by the framework, can be found in the sections ***Hair Coupe 1***, ***Hair Coupe 2***, ***Hair Coupe 3***, and ***Hair Coupe 4***, which are all located in ***Appendix C***. Every section contains a screen shot of the rendered image, and contains three tables listing the mechanical and visual properties of the hair coupe, and the collision geometry present in the scene.

- **Hair Coupe 1** contains a short black coloured hair coupe. The Super-Helix which determines the shape of the hair strand consists of one element of one centimetre long. Because the hair strand is very short, the internal elastic energy is dominating the potential gravitational energy resulting in a straight spiky hair.

- **Hair Coupe 2** contains a half long brown coloured hair coupe. The Super-Helix which determines the shape of the hair strand consists of four element of each two centimetre long. Because the hair strand is long, the potential gravitational energy is dominating the internal elastic energy resulting in hanging hair. The large helix step results into a straight hair style.

- **Hair Coupe 3** contains a long brown coloured hair coupe. The Super-Helix which determines the shape of the hair strand consists of four element of each four centimetre long. Because the hair strand is long, the potential gravitational energy is dominating the internal elastic energy resulting in hanging hair. The small helix step and small helix radius results in small waves in the hair style. To prevent hair from hanging before the eyes, a collision sphere is placed in front of the head.

- **Hair Coupe 4** contains a long blond coloured hair coupe. The Super-Helix which determines the shape of the hair strand consists of four element of each four centimetre long. Because the hair strand is long, the potential gravitational energy is dominates the internal elastic energy resulting in hanging hair. The small helix step and large helix radius results in large waves in the hair style. To prevent hair from hanging before the eyes, a collision sphere is placed in front of the head.

The quality of the rendered hair coupe depends largely on the resolution of the display device and the amount of hairs rendered. Rendering an image with a high hair density will increase the quality and realism of the image, but it will also increase the amount of time and memory consumed within the hair generation process.

5. Conclusions

During the research project a framework has been developed that allows the simulation and rendering of physical and visual believable human hair. This framework has been created by finding the solutions to the different sub problems found in the hair rendering and hair simulation, and combining these solutions into a single application.

A Super-Helix model – presented in (Bertails, et al., 2005b) – combined with Rosenbrock optimization method – presented in (Rosenbrock, 1960) – results in a physical correct model which can simulate the overall shape of a hair coupe when at rest. This model takes into account, both, the non-linear elastic properties of an individual hair strand and the gravitation field pulling at the hair strand.

Once the hair coupe mesh is generated it will be rendered by Kajiya & Kay's illumination model – presented in (Kajiya, et al., 1989); Kajiya & Kay's illumination model is a simple and effective hair illumination model. Shadows are added by using Deep Opacity Maps – presented in (Yuksel, et al., 2008); Deep Opacity Maps are superior in quality over other shadow mapping techniques when it comes to shadowing volumetric objects. The rendering is done in three passes and will result in the hair coupe being rendered to display.

A user of the framework will be able to specify the hair and environment properties which will influence the final shape and appearance of the hair coupe rendered. He has control over the following functionalities:

- **Hair creation** (selection of hair type and hairstyle)
- **Hair physics** (influence of gravity, elasticity and collision)
- **Hair shading** (presence of lighting, shadowing and colouring)

Specifying above properties allows a wide variety of hair shapes and hair appearances to be simulated and rendered by the framework. The quality of the rendered images depends largely on the resolution of the display device and the amount of hairs rendered.

To end the research project, the list below contains possible improvements and recommendations for the framework:

- **Implementing Interactive Tools**. Implementing interactive hairdresser tools, such as cutting, brushing and hair cosmetic, allows the user of the framework to add further details to the generated hair coupe.
- **Implementing Linear Super-Helices Model**. Implementing the Linear Super-Helix model, described in (Bertails, 2009) , allows the simulation of dynamic hair behaviour.
- **Implementing Marschner Illumination Model**. Implementing the Marschner Illumination model, described in (Marschner, et al., 2003) results in optical correct hair illumination, at the cost of a lower frame rate.
- **Exploiting the parallel power of GPUs**. Some equations in the hair framework can be accelerated by exploiting the computation power of the GPU. By performing General-purpose computing on GPU (GPGPU), the framework can use shader programs to perform parallel computation power of modern-day GPUs.

Appendices

Appendix A

Hair Shader

Vertex

```
1.
2.      uniform vec4 LightPosition;
3.      varying vec4 TexCoord;
4.
5.      varying vec3 tangent;
6.      varying vec3 light;
7.      varying vec3 eye;
8.
9.      void main()
10.     {
11.         vec3 vertex;
12.
13.         // Projective Texturing
14.         TexCoord = gl_TextureMatrix[0] * gl_Vertex;
15.
16.         // Tangent, Vertex, Light & Eye Passing
17.         tangent = vec3(gl_NormalMatrix    * gl_Normal);
18.         vertex  = vec3(gl_ModelViewMatrix * gl_Vertex);
19.         light   = vec3(gl_ModelViewMatrix * LightPosition) - vertex;
20.
21.         eye     = -vertex;
22.
23.         // Vertex transformation
24.         gl_Position = ftransform();
25.     }
26.
```

Fragment

```
1.
2.      uniform sampler2D ShadowMap;
3.      uniform sampler2D OpacityMap;
4.
5.      uniform vec4  LayerSize;
6.
7.      varying vec4 TexCoord;
8.
9.      uniform vec3 AColor;
10.     uniform vec3 DColor;
11.     uniform vec3 SColor;
12.
13.     uniform float Glossiness;
14.
15.     varying vec3 tangent;
16.     varying vec3 light;
17.     varying vec3 eye;
18.
19.     void main()
```

```
20.     {
21.         // Texture Fetch
22.         vec4 ShadowValue  = texture2DProj(ShadowMap, TexCoord);
23.         vec4 OpacityValue = texture2DProj(OpacityMap,TexCoord);
24.
25.         // Normalize Vectors
26.         vec3 T = normalize(tangent);
27.         vec3 L = normalize(light);
28.         vec3 E = normalize(eye);
29.
30.         // Calculate Angles
31.         float cosTL = abs(dot(T,L));
32.         float cosTE = abs(dot(T,E));
33.
34.         float sinTL = sin(acos(cosTL));
35.         float sinTE = sin(acos(cosTE));
36.
37.         // Calculate Intensities
38.         float ALight = 0.2;
39.         float DLight = sinTL;
40.         float SLight = pow(cosTL*cosTE + sinTL*sinTE, Glossiness);
41.
42.         // Calculate Shadows
43.         LayerSize /= (TexCoord.q*TexCoord.q);
44.         TexCoord  /= (TexCoord.q);
45.
46.         vec4 zi;
47.         zi.x = ShadowValue.x + LayerSize.x;
48.         zi.y = zi.x          + LayerSize.y;
49.         zi.z = zi.y          + LayerSize.z;
50.         zi.w = zi.z          + LayerSize.w;
51.
52.         float Shadow = 0.0;
53.
54.         vec4 mixer = 1.0 - (zi-TexCoord.p)/LayerSize;
55.
56.         if      (TexCoord.p < zi.x)
57.             Shadow = mix(0.0,            OpacityValue.r, mixer.x);
58.         else if (TexCoord.p < zi.y)
59.             Shadow = mix(OpacityValue.r, OpacityValue.g, mixer.y);
60.         else if (TexCoord.p < zi.z)
61.             Shadow = mix(OpacityValue.g, OpacityValue.b, mixer.z);
62.         else if (TexCoord.p < zi.z)
63.             Shadow = mix(OpacityValue.b, OpacityValue.a, mixer.w);
64.         else
65.             Shadow = OpacityValue.a;
66.
67.
68.         Shadow = 1.0 - (Shadow * 0.75);
69.
70.         // Calculate Final Colour
71.         vec3 Color =
72.             ALight * AColor +
73.             DLight * DColor * Shadow +
74.             SLight * SColor * Shadow * Shadow;
75.
76.         gl_FragColor = vec4(Color, 1.0);
77.     }
78.
```

Head Shader

Vertex

```
1.
2.      uniform vec4 LightPosition;
3.      varying vec4 TexCoord;
4.
5.      varying vec3 normal;
6.      varying vec3 light;
7.      varying vec3 eye;
8.
9.      void main()
10.     {
11.         vec3 vertex;
12.
13.         // Projective Texturing
14.         TexCoord = gl_TextureMatrix[0] * gl_Vertex;
15.
16.         // Normal, Vertex, Light & Eye Passing
17.         normal = vec3(gl_NormalMatrix    * gl_Normal);
18.         vertex = vec3(gl_ModelViewMatrix * gl_Vertex);
19.         light  = vec3(gl_ModelViewMatrix * LightPosition) - vertex;
20.
21.         eye    = -vertex;
22.
23.         // Vertex transformation
24.         gl_Position = ftransform();
25.     }
26.
```

Fragment

```
1.
2.      uniform sampler2D ShadowMap;
3.      uniform sampler2D OpacityMap;
4.
5.      uniform vec4 LayerSize;
6.
7.      varying vec4 TexCoord;
8.
9.      uniform vec3 AColor;
10.     uniform vec3 DColor;
11.
12.     varying vec3 normal;
13.     varying vec3 light;
14.     varying vec3 eye;
15.
16.     void main()
17.     {
18.         // Texture Fetch
19.         vec4 ShadowValue  = texture2DProj(ShadowMap, TexCoord);
20.         vec4 OpacityValue = texture2DProj(OpacityMap,TexCoord);
21.
22.         // Normalize Vectors
23.         vec3 N = normalize(normal);
24.         vec3 L = normalize(light);
```

```
25.         vec3 E = normalize(eye);
26.
27.         // Calculate Intensities
28.         float ALight  = 0.2;
29.         float DLight  = max(dot(N,L), 0.0);
30.
31.         // Calculate Shadows
32.         LayerSize /= (TexCoord.q*TexCoord.q);
33.         TexCoord  /= (TexCoord.q);
34.
35.         vec4 zi;
36.         zi.x = shadowValue.x + LayerSize.x;
37.         zi.y = zi.x          + LayerSize.y;
38.         zi.z = zi.y          + LayerSize.z;
39.         zi.w = zi.z          + LayerSize.w;
40.
41.         vec4 mixer = (1.0 - (zi-TexCoord.p)/LayerSize);
42.         float Shadow = 0.0;
43.
44.         if      (TexCoord.p < zi.x)
45.             Shadow = mix(0.0,            OpacityValue.r, mixer.x);
46.         else if (TexCoord.p < zi.y)
47.             Shadow = mix(OpacityValue.r, OpacityValue.g, mixer.y);
48.         else if (TexCoord.p < zi.z)
49.             Shadow = mix(OpacityValue.g, OpacityValue.b, mixer.z);
50.         else if (TexCoord.p < zi.w)
51.             Shadow = mix(OpacityValue.b, OpacityValue.a, mixer.w);
52.         else
53.             Shadow = OpacityValue.a;
54.
55.         Shadow = 1.0 - (Shadow * 0.75);
56.
57.         // Calculate Final Colour
58.         vec3 Color =
59.             ALight * AColor +
60.             DLight * DColor * Shadow;
61.
62.         gl_FragColor = vec4(Color, 1.0);
63.     }
64.
```

Opacity Shader

Vertex

```
1.
2.      varying vec4 TexCoord;
3.
4.      void main()
5.      {
6.          // Projective Texturing
7.          TexCoord = gl_TextureMatrix[0] * gl_Vertex;
8.
9.          // Vertex transformation
10.         gl_Position = ftransform();
11.     }
```

Fragment

```
1.
2.      uniform sampler2D ShadowMap;
3.
4.      varying vec4 TexCoord;
5.
6.      uniform float Opacity;
7.      uniform vec4  LayerSize;
8.
9.      void main()
10.     {
11.         vec4 Color;
12.
13.         // Texture Fetch
14.         vec4 ShadowValue = texture2DProj(ShadowMap, TexCoord);
15.
16.         // Calculate Opacity
17.         LayerSize /= (TexCoord.q*TexCoord.q);
18.         TexCoord  /= (TexCoord.q);
19.
20.         vec4 zi;
21.         zi.x = shadowValue.x + LayerSize.x;
22.         zi.y = zi.x          + LayerSize.y;
23.         zi.z = zi.y          + LayerSize.z;
24.         zi.w = zi.z          + LayerSize.w;
25.
26.         if (TexCoord.p < zi.x)
27.         {
28.             Color.r = Opacity;
29.             Color.g = Opacity;
30.             Color.b = Opacity;
31.             Color.a = Opacity;
32.         }
33.         else if (TexCoord.p < zi.y)
34.         {
35.             Color.g = Opacity;
36.             Color.b = Opacity;
37.             Color.a = Opacity;
38.         }
39.         else if (TexCoord.p < zi.z)
40.         {
41.             Color.b = Opacity;
42.             Color.a = Opacity;
43.         }
44.         else
45.         {
46.             Color.a = Opacity;
47.         }
48.
49.         gl_FragColor = Color;
50.     }
51.
```

Appendix B

Input Files

Setup

```
1.
2.      ########################################
3.      ## Setup File                          ##
4.      ########################################
5.
6.      # Load & Save
7.
8.      Load            true
9.      Save            false
10.     LoadFile        Hair/Hair1.hair
11.     SaveFile        Hair/Hair1.hair
12.
13.     # Atributes
14.
15.     HairstyleInfo   Properties/Hairstyle/Hairstyle1.txt
16.     CollisionInfo   Properties/Collision/Collision1.txt
17.     VisualityInfo   Properties/Visuality/Visuality1.txt
18.
19.     # Head
20.
21.     HeadModel       Assets/Objects/Head.obj
22.     HeadVert        Assets/Shaders/Phong_Shadow.vert
23.     HeadFrag        Assets/Shaders/Phong_Shadow.frag
24.
25.     # Hair
26.
27.     WickModel       Assets/Objects/Wick.obj
28.     HairVert        Assets/Shaders/KajiyaKay_Shadow.vert
29.     HairFrag        Assets/Shaders/KajiyaKay_Shadow.frag
30.
```

Hairstyle Info

```
1.
2.      ########################################
3.      ## Hairstyle File: Medium Straight    ##
4.      ########################################
5.
6.      # Precision
7.
8.      VisualityStep       0.0025
9.      CollisionStep       0.0050
10.
11.     # Minimization
12.
13.     Maxiter             5
14.     Epsilon             0.0001
15.
16.     # Hair Shape
17.
18.     HairLengths         0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
```

```
19.    HairDensity          80
20.
21.    # Elasticity
22.
23.    HairRadius1          0.000035
24.    HairRadius2          0.000050
25.
26.    HelixRadius          0.050
27.    HelixStep            0.050
28.
29.    Poisson              0.48
30.    Young                1000000000
31.
32.    # Gravity
33.
34.    Density              13000
35.    Gravity              9.81
36.
```

Collision Info

```
1.
2.     ########################################
3.     ## Collision File                     ##
4.     ########################################
5.
6.     # Elipsoid [Volume:1] [Penalty:1] [Centre:3] [Dimensions:3] [Axis:9]
7.     # Cuboid   [Volume:1] [Penalty:1] [Centre:3] [Dimensions:3] [Axis:9]
8.     # Sphere   [Volume:1] [Penalty:1] [Centre:3] [Radius:1]
9.
10.    # Collision List
11.
12.    Elipsoid 0.04  10.00  0.00 0.07 0.01  0.08 0.08 0.11  1 0 0 0 1 0 0 0 1
13.    Sphere  -0.02   0.01  0.00 0.04 0.16  0.10
14.
```

Visuality Info

```
1.
2.     ########################################
3.     ## Visuality File: Brown               ##
4.     ########################################
5.
6.     # Blond              0.863  0.549  0.000
7.     # Brown              0.471  0.235  0.000
8.     # Black              0.039  0.039  0.000
9.
10.    # Visualization
11.
12.    AmbientColor         0.392  0.584  0.929
13.    DiffuseColor         0.471  0.235  0.000
14.    SpecularColor        0.200  0.200  0.200
15.
16.    Glossiness           60
17.
```

Appendix C

Hair Coupe 1

Figure 38. Hair Coupe 1

Table 5. Hair Coupe 1 - Collision Properties

Collision	Volume	Penalty	Centre	Dimensions	Axis
Ellipsoid	0.00	10.0	-0.001 +0.068 +0.010	0.080 0.080 0.110	1 0 0 0 1 0 0 0 1

Table 6. Hair Coupe 1 - Hairstyle Properties

Hairstyle			
Visuality Step	0.0025	HairRadius1	0.000035
Collision Step	0.0050	HairRadius2	0.000050
Hair Lengths	0.01		
Hair Density	750	Young	1000000000
Helix Radius	0.002	Poisson	0.48
Helix Step	0.005		
MaxIter	20	Density	13000
Epsilon	0.0001	Gravity	9.81

Table 7. Hair Coupe 1 - Visual Properties

Visuality			
Ambient Colour	0.10 0.10 0.10	Glossiness	60
Diffuse Colour	0.00 0.00 0.00		
Specular Colour	0.50 0.50 0.50		

Hair Coupe 2

Figure 39. Hair Coupe 1

Table 8. Hair Coupe 2 - Collision Properties

Collision	Volume	Penalty	Centre	Dimensions	Axis
Ellipsoid	0.00	10.0	-0.001 +0.068 +0.010	0.080 0.080 0.110	1 0 0 0 1 0 0 0 1

Table 9. Hair Coupe 2 - Hairstyle Properties

Hairstyle			
Visuality Step	0.0025	HairRadius1	0.000035
Collision Step	0.0050	HairRadius2	0.000050
Hair Lengths	0.02 0.02 0.02 0.02		
Hair Density	100	Young	1000000000
Helix Radius	0.050	Poisson	0.48
Helix Step	0.050		
MaxIter	20	Density	13000
Epsilon	0.0001	Gravity	9.81

Table 10. Hair Coupe 2 - Visual Properties

Visuality			
Ambient Colour	0.10 0.10 0.10	Glossiness	60
Diffuse Colour	0.45 0.25 0.00		
Specular Colour	0.20 0.20 0.20		

Hair Coupe 3

Figure 40. Hair Coupe 3

Table 11. Hair Coupe 3 - Collision Properties

Collision	Volume	Penalty	Centre	Dimensions	Axis
Ellipsoid	0.04	10.0	-0.001 +0.068 +0.010	0.080 0.080 0.110	1 0 0 0 1 0 0 0 1
Sphere	0.15	0.01	+0.000 +0.033 +0.165	0.080	N/A

Table 12. Hair Coupe 3 - Hairstyle Properties

Hairstyle			
Visuality Step	0.0025	HairRadius1	0.000035
Collision Step	0.0050	HairRadius2	0.000050
Hair Lengths	0.04 0.04 0.04 0.04		
Hair Density	80	Young	1000000000
Helix Radius	0.025	Poisson	0.48
Helix Step	0.025		
MaxIter	20	Density	13000
Epsilon	0.0001	Gravity	9.81

Table 13. Hair Coupe 3 - Visual Properties

Visuality			
Ambient Colour	0.10 0.10 0.10	Glossiness	60
Diffuse Colour	0.45 0.25 0.00		
Specular Colour	0.20 0.20 0.20		

Hair Coupe 4

Figure 41. Hair Coupe 4

Table 14. Hair Coupe 4 - Collision Properties

Collision	Volume	Penalty	Centre	Dimensions	Axis
Ellipsoid	0.00	10.0	-0.001 +0.068 +0.010	0.080 0.080 0.110	1 0 0 0 1 0 0 0 1
Sphere	0.05	0.01	+0.000 +0.040 +0.150	0.095	N/A

Table 15. Hair Coupe 4 - Hairstyle Properties

Hairstyle			
Visuality Step	0.0025	HairRadius1	0.000035
Collision Step	0.0050	HairRadius2	0.000050
Hair Lengths	0.04 0.04 0.04 0.04		
Hair Density	80	Young	1000000000
Helix Radius	4.000	Poisson	0.48
Helix Step	0.025		
MaxIter	20	Density	13000
Epsilon	0.0001	Gravity	9.81

Table 16. Hair Coupe 4 - Visual Properties

Visuality			
Ambient Colour	0.10 0.10 0.10	Glossiness	60
Diffuse Colour	0.90 0.65 0.10		
Specular Colour	0.20 0.20 0.20		

Appendix D

Hardware

The hardware used in this project includes:

- An Asus notebook computer containing a Intel Centrino Core 2 Duo CPU running at a speed of 2.00 GHz and containing 2.00 GB of RAM memory. The notebook contains an Nvidia GeForce 8400m G GPU with 128 MB memory and runs Microsoft Windows 7 as its operating system. The display of the notebook has its resolution set to 1280 by 800 pixels.
- A desktop computer containing a Intel Core 2 Duo CPU running at a speed of 2.40 GHz and containing 4.00 GB of RAM memory. The desktop computer contains an Nvidia GeForce 8800 GTS with 640 MB memory and runs Microsoft Windows 7 as its operating system. The display of the desktop computer has its resolution set to 1280 by 1024 pixels.

The Asus notebook's main purpose in this project is to develop the application that generates animations and test the code implementation. It is also used to test and validate the models, algorithms and techniques used within the application. The desktop computer is used for screen capturing the demonstration videos, and verifying whether the framework is portable.

Software

The software used in this project includes:

- Java 1.6 Programming Language
- Netbeans IDE 6.8
- JOGL 1.0 and GLSL

Java 1.6 is the programming language used for implementing the hair framework. The reason for choosing Java over other languages is that it is a robust, generic and a simple to understand object-oriented programming language, allowing great flexibility in the way how the framework is implemented. An additional advantage for using Java in this project is that it will be simpler to incorporate parts of the framework into existing software projects developed at the University of Twente, as a majority of the code base present at the University of Twente is written in Java.

Netbeans IDE 6.8 is used in this project, because it provides an easy to use integrated development environment, which provides insight into the structure and the development of the hair framework. Importing the Netbeans Java OpenGL Pack Plug-In provides an easy to use OpenGL development environment integrated into the Netbeans IDE.

Java OpenGL (JOGL) is a wrapper library that allows OpenGL to be used in the Java programming language. JOGL allows access to most features available to C programming language programmers, with the notable exception of window-system related calls in GLUT (as Java contains its own windowing systems, AWT and Swing), and some extensions.

Initially was chosen to use JOGL 2 Beta – which supports OpenGL 3.1. However, because no functionality of OpenGL 3.1 was needed in the project and JOGL 2.0 Beta showed signs of incompletement, I have decided to use JOGL 1.0 instead. Shaders are implemented in GL Shading Language (GLSL).

Bibliography

Ando Makoto en Morishima Shigeo Expression and Motion Control of Hair using Fast Collision Detection Methods [Conferentie] // Image Analysis Applications and Computer Graphics. - London : Springer-Verlag, 1995. - pp. 463-488.

Anjyo Ken-Ichi, Usami Yoshiaki en Kurihara Tsuneya A Simple Method for Extracting the Natural Beauty of Hair [Tijdschrift] // Computer Graphics. - 1992. - 2 : Vol. 26. - pp. 111-120.

Baalcke Mario Modelliering und Realistische Darstellung eines Menschlichen Kopfes [Rapport]. - Rostock : Institut für Informatik der Universitat Rostock, 2006.

Baker Will en King Scott Interactive Modelling of Hair with Texture Maps [Conferentie] // Image and Vision Computing. - Palmerston North : University of Otago, 2003. - pp. 84-89.

Bando Yosuke, Chen Bing-Yu en Nishita Tomoyuki Animating Hair with Loosely Connected Particles [Tijdschrift] // Eurographics. - 2003. - 3 : Vol. 22. - pp. 411-418.

Bertails Florence [et al.] Adaptive Wisp Tree - A Multiresolution Control Structure for Simulating Dynamic Clustering in Hair Motion [Conferentie] // Computer Animation. - San Diego : Eurographics Association, 2003. - pp. 207-213.

Bertails Florence [et al.] Predicting Natural Hair Shapes by Solving the Statics of Flexible Rods [Conferentie] // Eurographics. - Dublin : ACM, 2005b. - p. 1.

Bertails Florence [et al.] Realistic Hair Simulation - Animation and Rendering [Boek]. - Los Angeles : ACM, 2008.

Bertails Florence [et al.] Super-Helices for Predicting the Dynamics of Natural Hair [Conferentie] // Computer Graphics and Interactive Techniques. - Boston : ACM, 2006. - pp. 1180-1187.

Bertails Florence Linear Time Super-Helices [Conferentie] // Eurographics. - Oxford : Blackwell Publishing, 2009.

Bertails Florence Simulation de Chevelures Virtuelles [Rapport]. - Grenoble : Institut National Polytechnique de Grenoble, 2006.

Bertails Florence, Ménier Clément en Cani Marie-Paule A Practical Self-shadowing Algorithm for Interactive Hair Animation [Conferentie] // Graphics Interface. - Victoria : Canadian Human-Computer Communications Society, 2005a. - pp. 71-78.

Booth Andrew Numerical Methods [Boek]. - London : Butterworths, 1955. - pp. 95-100, 154-160.

Cani Marie-Paule en Bertails Florence Hair Interactions [Sectie van boek] // Collision Handling and its applications. - [sl] : The Eurographics Association, 2006.

Chang Johnny, Jin Jingyi en Yu Yizhou A Practical Model for Hair Mutual Interactions [Conferentie] // Computer Animation. - San Antonio : ACM, 2002. - pp. 73-80.

Chen Lieu-Hen [et al.] A System of 3D Hair Style Synthesis based on the Wisp Model [Tijdschrift] // The Visual Computer. - 1999. - 4 : Vol. 15. - pp. 159-170.

Choe Byoungwon, Gyu Choi Min en Ko Hyeong-Seok Simulating Complex Hair with Robust Collision Handling [Conferentie] // Computer Animation. - Los Angeles : ACM, 2005. - pp. 153-160.

Daldegan Agnes [et al.] An Integrated System for Modeling, Animating and Rendering Hair [Tijdschrift] // Computer Graphics Forum. - 1993. - 3 : Vol. 12. - pp. 211-221.

Fletcher R en Powell M A Rapidly Convergent Descent Method for Minimization [Tijdschrift] // Computer Journal. - Surray : British Computer Society, 1963. - Vol. 6. - pp. 163-168.

Grabli Stéphane [et al.] Image-Based Hair Capture by Inverse Lighting [Tijdschrift] // Graphics Interface. - 2002. - pp. 51-58.

Guang Yang en Zhiyong Huang A Method of Human Short Hair Modeling and Real Time Animation [Conferentie] // Computer Graphics and Applications. - Washington : IEEE Computer Society, 2002. - pp. 435-438.

Gupta Rajeev [et al.] Optimized Framework for Real Time Hair [Conferentie] // Computer Graphics International. - Berlin : Springer-Verlag, 2006. - pp. 702–710.

Gupta Rajeev en Magnenat-Thalmann Nadia Scatering-Based Interactive Hair Rendering [Conferentie] // Computer Aided Design and Computer Graphics. - Hong Kong : IEEE Computer Society, 2005. - pp. 489-496.

Hadap Sunil [et al.] Strands and Hair: Modeling, Animation, and Rendering [Boek]. - San Diego : ACM, 2007.

Hadap Sunil en Magnenat-Thalmann Nadia Modeling Dynamic Hair as a Continuum [Tijdschrift] // Eurographics. - 2001. - 3 : Vol. 20. - pp. 329–338.

Hadap Sunil en Magnenat-Thalmann. Nadia Interactive Hair Styler based on Fluid Flow [Tijdschrift] // Computer Animation and Simulation. - 2000. - pp. 87-100.

James Blinn Light Reflection Functions for Simulation of Clouds and Dusty Surfaces [Tijdschrift] // Computer Graphics. - 1982. - 3 : Vol. 16. - pp. 21-29.

Kajiya James en Kay Timothy Rendering Fur with Three Dimensional Textures [Tijdschrift] // Computer Graphics. - 1989. - 3 : Vol. 23. - pp. 271-280.

Kelly Ward [et al.] Modeling Hair Using Level-of-Detail Representations [Conferentie] // Computer Animation and Social Agents. - Chapel Hill : IEEE Computer Society, 2003. - pp. 41-47.

Kim Tae-Yong en Neumann Ulrich A Thin Shell Volume for Modeling Human Hair [Conferentie] // Computer Animation. - Philadelphia : IEEE Computer Society, 2000. - pp. 104-111.

Kim Tae-Yong en Neumann Ulrich Interactive Multiresolution Hair Modeling and Editing [Conferentie] // Computer graphics and interactive techniques. - San Antonio : ACM, 2002. - pp. 620-629.

Kmoch Petr Hair Animation Methods [Conferentie] // Doctoral Students - Part I: Mathematics and Computer Sciences . - Prague : Charles University, 2007. - pp. 34-39.

Koh Chuan Koon en Huang Zhiyong A Simple Physics Model to Animate Human Hair Modeled in 2D Strips in Real Time [Conferentie] // Eurographic Workshop on Computer Animation and Simulation. - Manchester : Springer-Verlag, 2001. - pp. 127-138.

Koh Chuan Koon en Zhiyong Huang Real-Time Animation of Human Hair Modeled in Strips [Conferentie] // Computer Animation and Simulation. - Wien : Springer-Verlag, 2000.

Kong Wai Ming, Takahashi Hiroki en Naka Masayuki Generation of 3D Hair Model from Multiple Pictures [Tijdschrift] // Multimedia Modeling. - 1997. - pp. 183-196.

Kong Waiming en Nakajima Masayuki Visible Volume Buffer for Efficient Hair Expression and Shadow Generation [Conferentie] // Computer Animation. - Washington : IEEE Computer Society, 1999. - pp. 58-65.

Koster Martin, Haber Jörg en Seidel Hans-Peter Real-Time-Rendering of Human Hair using Programmable Graphics Hardware [Conferentie] // Computer Graphics International. - Crete : IEEE Computer Society, 2004. - pp. 248-256.

Kurihara Tsuneya, Anjyo Ken-ichi en Thalmann Daniel Hair Animation with Collision Detection [Tijdschrift] // Models and Techniques in Computer Animation. - 1993. - pp. 128-138.

LeBlanc André, Turner Russell en Thalmann Daniel Rendering Hair using Pixel Blending and Shadow Buffers [Tijdschrift] // Visualization and Computer Animation. - 1991. - 3 : Vol. 2. - pp. 92-97.

Lee Do-Won en Ko Hyeong-Seok Natural Hairstyle Modeling and Animation [Tijdschrift] // Human Modeling and Animation. - Seoul : Korea Computer Graphics Society, 2000. - pp. 11-21.

Lengyel Jerome [et al.] Real-time Fur over Arbitrary Surfaces [Conferentie] // Interactive 3D Graphics. - New York : ACM, 2001. - pp. 227 - 232.

Liang Wenqi en Huang Zhiyong An Enhanced Framework for Real-time Hair Animation [Conferentie] // Computer Graphics and Applications. - Washington : IEEE Computer Society, 2003. - pp. 467-471.

Lokovic Tom en Veach Eric Deep Shadow Maps [Conferentie] // Computer Graphics. - [sl] : Addison-Wesley, 2000. - pp. 385-392.

Magnenat-Thalmann Nadia en Hadap Sunil State of the Art in Hair Simulation [Tijdschrift] // International Workshop on Human Modeling and Animation. - 2000. - pp. 3-9.

Mao Xiaoyang [et al.] Sketch Interface Based Expressive Hairstyle [Tijdschrift] // Computer Graphics International. - Washington : IEEE Computer Society, 2004. - pp. 608-611.

Marschner Stephen, Jensen Henrik en Cammarano Mike Light Scattering from Human Hair Fibers [Conferentie] // ACM Transactions on Graphics. - San Diego : ACM, 2003. - pp. 780-791.

Matt Pharr GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation [Boek] = GPU Gems 2. - New York : Addison-Wesley Professional, 2005.

McAdams Aleka [et al.] Detail Preserving Continuum Simulation of Straight Hair [Conferentie] // Computer Graphics and Interactive Techniques. - New Orleans : ACM, 2009. - pp. 1-6.

Mertens Tom [et al.] A Self-Shadow Algorithm for Dynamic Hair using Density Clustering [Conferentie] // Computer Graphics and Interactive Techniques. - Los Angeles : ACM, 2004. - pp. 44-49.

Moon Jonathan en Marschner Stephen Simulating multiple scattering in hair using a photon mapping approach [Conferentie] // Computer Graphics and Interactive Techniques. - Boston : ACM, 2006. - pp. 1067-1074 .

Moon Jonathan, Walter Bruce en Marschner Steve Efficient Multiple Scattering in Hair using Spherical Harmonics [Conferentie] // Computer Graphics and Interactive Techniques. - Los Angeles : ACM, 2008. - pp. 1-7.

Noble Paul en Tang Wen Modelling and Animating Cartoon Hair with Nurbs Surfaces. [Tijdschrift]. - Crete : Teeside University, 2004. - pp. 60-67.

Palmer John An Improved Procedure for Orthogonalising the Search Vectors in Rosenbrock's and Swann's Direct Search Optimization Methods [Tijdschrift] // The Computer Journal. - Surrey : British Computer Society, 1969. - 12 : Vol. 1969. - pp. 69-71.

Paris Sylvain, Briceño Hector en Sillion François Capture of Hair Geometry from Multiple Images [Tijdschrift] // Transactions on Graphics. - 2004. - pp. 712-719.

Patrick Deborah, Bangay Shaun en Lobb Adele Modelling and Rendering Techniques for African Hairstyles [Conferentie] // Computer graphics, Virtual Reality, Visualisation and Interaction in Africa,. - Stellenbosch : ACM Press, 2004. - Vol. 3. - pp. 115-124.

Petrovic Lena, Henne Mark en Anderson John Volumetric Methods for Simulation and Rendering of Hair [Rapport]. - Emeryville : Pixar Animation Studios, 2005.

Plante Eric, Cani Marie-Paule en Poulin Pierre A Layered Wisp Model for Simulating Interaction inside Long Hair [Conferentie] // Eurographic workshop on Computer animation and simulation. - New York : Springer-Verlag, 2001. - pp. 139-148.

Reeves William Particle Systems — a Technique for Modeling a Class of Fuzzy Objects [Tijdschrift] // ACM Transactions on Graphics. - 1983. - 2 : Vol. 2. - pp. 91-108.

Robbins Clarence Chemical and Physical Behaviour of Human Hair [Boek]. - New York : Springer-Verlag, 2002.

Robertson Barbara Hair-Raising Effects [Artikel] // Computer Graphics World. - October 1995.

Rosenblum Robert, Carlson Wayne en Tripp Edwin Simulating the Structure and Dynamics of Human Hair - Modelling, Rendering and Animation [Tijdschrift] // Visualization and Computer Animation. - 1991. - 4 : Vol. 2. - pp. 141-148.

Rosenbrock Howard An Automatic Method for Finding the Greatest or Least Value of a Function [Tijdschrift]. - Surrey : British Computer Society, 1960. - Vol. 3. - pp. 175-184.

Scheuermann Thorsten Practical Real-Time Hair Rendering and Shading [Conferentie] // Computer Graphics and Interactive Techniques. - Los Angeles : ACM, 2004. - p. 147.

Selle Andrew, Lentine Michael en Fedkiw Ronald A Mass Spring Model for Hair Simulation [Tijdschrift] // Transactions on Graphics. - 2008. - 3 : Vol. 27. - pp. 1-11.

Sintorn Erik en Assarsson Ulf Real-Time Approximate Sorting for Self Shadowing and Transparency in Hair [Conferentie] // Interactive 3D graphics and games. - Redwood City : ACM, 2009. - pp. 157-162.

Sobottka Gerrit en Weber Andreas Computing Static Electricity on Human Hair [Tijdschrift] // Theory and Practice of Computer Graphics. - 2006. - pp. 21–29.

Sokol Anna Modeling Hair Movement with Mass-Springs [Rapport]. - Stony Brook : Stony Brook University, 2003.

Volino Pascal en Magnenat-Thalmann Nadia Animating Complex Hairstyles in Real-Time [Conferentie] // Virtual Reality Software and Technology. - Hong Kong : ACM, 2004. - pp. 41-48.

Wang Lvdi [et al.] Examplebased Hair Geometry Synthesis [Conferentie] // SIGGRAPH. - New Orleans : ACM, 2009. - Vol. 22. - pp. 1-9.

Ward Kelly [et al.] A Survey on Hair Modeling - Styling, Simulation, and Rendering [Tijdschrift] // Visualization and Computer Graphics. - 2007a. - 2 : Vol. 13. - pp. 213-234.

Ward Kelly en Lin Ming Adaptive Grouping and Subdivision for Simulating Hair Dynamics [Conferentie] // Computer Graphics and Applications. - Chapel Hill : IEEE Computer Society, 2003. - pp. 234- 243.

Ward Kelly, Galoppo Nico en Ming Lin A Simulation-based VR System for Interactive Hairstyling [Conferentie] // Virtual Reality. - Washington : IEEE Computer Society, 2006. - pp. 257-260.

Ward Kelly, Galoppo Nico en Ming Lin Interactive Virtual Hair Salon [Tijdschrift] // Presence: Teleoperators and Virtual Environments. - 2007b. - 3 : Vol. 16. - pp. 237-251.

Watanabe Yasuhiko en Suenaga Yasuhito Drawing Human Hair using the Wisp Model [Tijdschrift] // The Visual Computer. - 1991. - 2-3 : Vol. 7. - pp. 97-103.

Wei Yichen [et al.] Modeling Hair from Multiple Views [Tijdschrift] // Graphics. - 2005. - pp. 816-820.

Yang Tzong-Jer en Ouhyoung Ming Rendering Hair with Back-Lighting [Conferentie] // CAD/Graphics. - Shenzhen : National Taiwan University, 1997. - pp. 291-296.

Yu Yizhou Modeling Realistic Virtual Hairstyles [Conferentie] // Pacific Conference on Computer Graphics and Applications. - Washington : IEEE Computer Society, 2001. - pp. 295-304.

Yuksel Cem en Keyser John Deep Opacity Maps [Tijdschrift] // Computer Graphics Forum. - 2008. - 2 : Vol. 27. - pp. 1-6.

Yuksel Cem, Akleman Ergun en Keyser John Practical Global Illumination for Hair Rendering [Conferentie] // Computer Graphics and Applications . - Maui : IEEE Computer Society, 2007. - pp. 415-418.

Zhang Rui en Wünsche Burkhard A Framework for Interactive GPU-Supported Rendering and Styling of Virtual Hair [Conferentie] // Computer Graphics Theory and Applications. - Barcelona : INSTICC, 2007. - pp. 204-211.

Zinke Arno [et al.] Dual Scattering Approximation for Fast Multiple Scattering in Hair [Conferentie] // Computer Graphics and Interactive Techniques. - Los Angeles : ACM, 2008. - pp. 1-10.

Zinke Arno, Sobottka Gerrit en Weber Andreas Photo-Realistic Rendering of Blond Hair [Conferentie] // Vision, Modeling, and Visualization. - Stanford : IOS Press, 2004. - pp. 91-98.

Index