Lighting and Creating Computer Generated Hair

Masters Thesis

Gerard keating BA PG Dip

N.C.C.A Bournemouth University

September 10, 2007

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The creation of photo realistic or visually appealing simulated hair is one of the most difficult endeavours in modern computer graphics today. The success of some modern motion pictures has actually depended in part on the simulation of aesthetically pleasing hair. Movies such as the groundbreaking Stuart Little[SL199] and Stuart Little 2[SL202] and the more recent King Kong[Kon05] had their main characters covered in simulated short hair. Films such as Final Fantasy: the Spirits within[ff001] and The Incredibles[Inc04] had simulated human characters with long dynamic hair. Therefore even though hair simulation is difficult it has been fundamental to the success of the above films. The difficulties of hair simulation according to[MTHK02] can be broken down into three large areas: hair modeling, hair rendering and hair dynamics/animation. As opposed to solving one particular problem this project looked into the many issues that occur in hair model ling and hair rendering but due to time constraints hair dynamics/animation were not researched but left for future work.

This project was originally conceived to focus primarily on hair rendering with some references to a previous project [Kea07] on hair model ling created by the author. While researching this master thesis superior interpolation and model ling techniques were found. Also during the implementation and testing of the hair shading, deficiencies were found in [Kea07] and these will be discussed in *Chapter 4* and *Chapter 5*.

This project focuses on production ready hair rendering and creation. Hence production ready software is used. The project also presents a production pipeline that was used to create the images and videos in this project. This is presented in *Chapter 3*.

The two greatest challenges facing hair rendering is the unique nature of hair and the sheer amount of hair a regular character will have. Hair can be viewed as infinitesimally thin tubes since the diameter of human hair ranges from 17 to 181μm (millionths of a meter) hence it can described as rendering

one dimensional models in a three dimensional world whereas most rendering algorithms focus on the rendering of polygonal surfaces. As will be seen in *Chapter 2* real hair is far more complicated then this . Rendering and creating shaders for such complex geometry is covered in *Chapter 6*.

Modeling this type of geometry is different then modeling surfaces. An artist will never be able to model every individual strand of hair by hand so techniques have been developed to allow them to create and control all these hair strands. These techniques and an example system are described in *Chapter 4*.

The human head has approximately 100000 to 150000 hairs and many creatures have more then this covering their body. Even with the simplistic nature of individual hairs this can lead to long render times. In Final Fantasy: the Spirits within[ff001] approximately 25% of the rendering time was used for the main character's hair[1] and furthermore ninety percent of the memory used for storing the main characters geometry contained her hair[Bjo01]. Hence hair is expensive both in terms of computations and memory. The memory can be reduced using a render-time interpolation system such as the one implemented for this project which will be discussed in *Chapter 5* and this can also improve render times since there is less disk access.

1.2 Previous Work

The issue of rendering photo-realistic hair has been a concern of computer graphics research for long time. [CHP+79] is often considered to be the first attempt at rendering such complex geometry. The first major break through in the area came with [KK89]. The specular shading model created in that paper is still the basis for most modern hair shaders including the one presented in this project. It is discussed in more detail in Section 6.1. The model has been criticized and superseded in [MJC+03] which presents a shading model that is based on actual physical recordings of how light interacts with hair. The results of theses findings are discussed in Section 2.3. Research done on real hair came from medical anatomy books such as [Cre92]. An even better source for information about hair can be found in forensic research such as [Rob99]. As stated in [HDK+06], Marschener's results can be "faked" without using the ray tracing model he proposes. These fakes allow the artist more intuitive control over the final image. These are discussed in Section 6.1. Hair self shadowing is an important visual aspect of actual hair. Even though this can be achieved using standard depth based shadowing[RSC87], the deep shadowing technique described in [LV00] greatly enhances and speeds up the process as described in Section 6.3.

The first major attempt of photo-realistic hair in production was Stuart Little[SL199]. The pipeline and techniques used on this film is explained in [Bre00] and is the major inspiration for the pipeline for this project. Many more films, commercials and even television shows now have computer generated hair

[1] Actually 30% of all render cycles where for the main character Aki and 80% of her render time was dedicated to her hair which works out to be approximately 25%.

in them. Productions of note are Final Fantasy: the Spirits within [ff001]. The description of their hair system can be found in various articles, the most useful description can be found in [Bjo01]. The explanation of how the hairy creatures were rendered in [nar05] in [HDK+06] were also invaluable. The use of "visual programming" at Weta Digital for [Kon05] hinted at in [PH06] for creating even dirty hair is the current bleeding edge of hair creation and rendering. Even though it was not implemented in this project it does merit future research.

Chapter 2

Real Hair

2.1 The Anatomy of a Hair Strand

Before trying to replicate any existing natural objects one should study real objects so that measurements of how 'real' it looks can be made. Many animators and artists study subjects such as anatomy to better interpret the human or animal form. Another motivation for studying real hair is that many early renders of hair appeared like synthetic hair. To counteract, this research into real must be done to find the difference. Aside from phenomenological research, including collecting human hair clippings and handling pet rats, the author looked into scientific research done in the area.

Even though many shading models assume hair is perfectly round in cross section, it is in fact more elliptical and irregular as can be seen in figure2.1.

Furthermore, according to [Cre92], how elliptical or 'flat' hair is, is proportional to how curly the hair appears. Even though this property of hair can only be seen under a microscope, it does mean that hair looks different when viewed from different angles.

The outer layer of the hair known as the cuticle is not smooth but scaly as shown in figure 2.2. These overlapping scales have a surface tilted towards the root end of the hair by an average amount of three degrees according to [MJC+03]. [MJC+03] uses this microscopic scaling to explain part of the strange specular properties of hair.

The pigment of hair is mainly caused by the presence of pigment granules in the hair shaft[OT87]. Dark hair has a large amount of these granules, red hair has a medium amount, blonde hair has a little amount and white has very little to no pigment in it. In terms of hair shading this makes lighter hair more difficult to accurately render in a photo-realistic way since the colour is due more to light that passes through the hair or light that is internally reflected. These are more difficult to calculate then reflective rays which contribute most of the lighting of darker hairs.

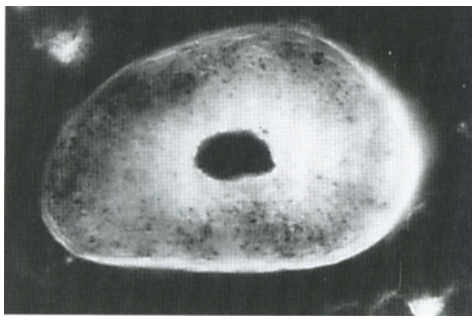

Figure 2.1: A cross section of a human pubic hair showing the elliptical nature of hair and the scattering of pigment granules which is quite dense here, except for the very centre known as the medulla. Taken from [Rob99]

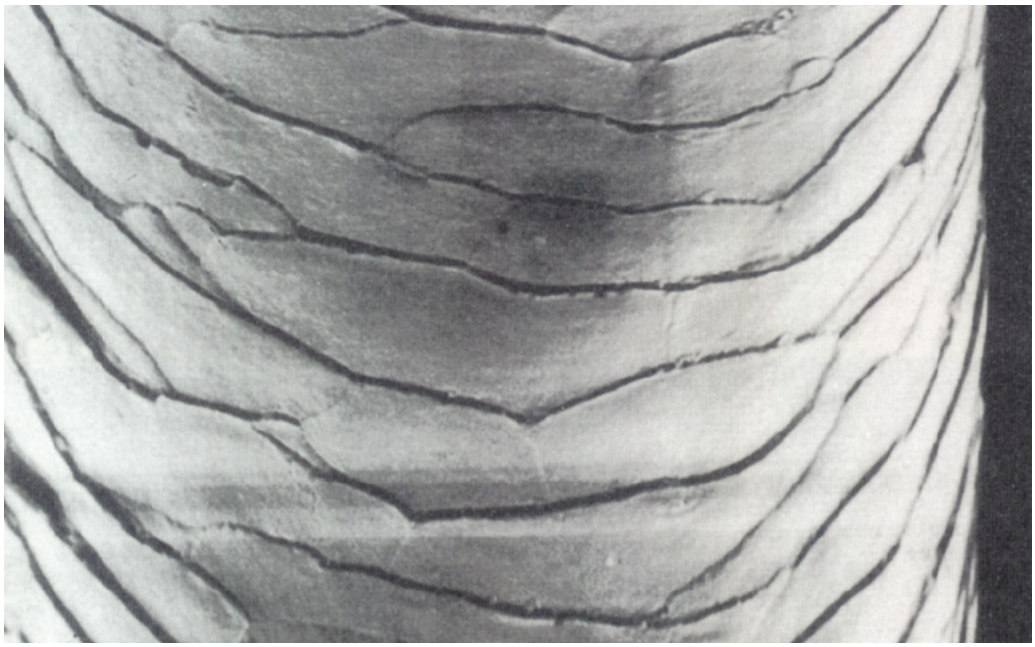

Figure 2.2: An image taken by an electron microscope of a hair showing overlapping scales of a hair shaft. The image is orientated so that the root of the hair is at the bottom of the image and the tip of the hair is at the top. It was taken from [MJC+03]

2.2 The Difference between Human and Animal Hair

The main difference between humans and other mammals is the lack visible hair on most humans in comparison to many other mammals even though there are exceptions such as whales which have no hair follicles. Non-human hairs usually have more variation in their hair colour from hair to hair.

One can visibly see guard hairs and under hairs in fur or peltage. The former being longer, coarser and less frequent then the finer under-hairs. Of course every creature has certain unique properties to their hair which would have to be studied on a creature to creature basis.

2.3 Dual Highlights

In [MJC+03] they point out that the Kajiya, Kay model used in this project and explained in Section 6.1, is a phenomenological model, not based on any scientific recording of how light interacts with hair strands. In creating their own hair lighting model they ran tests to record exactly how light interacts with hair strands. The method of finding the results can be seen in the paper but here we only present the findings.

Two specular highlights exist, a primary and secondary one. This is due to the reflectance from the surface of the hair and the internal reflectance of light. Internal reflectance is caused by light entering the hair strand because hair is semi-transparent and then being internally reflected back out.

As was said in Section 2.1 hair is not a smooth surface so this second reflectance is at a different angle then the first this means that for real hair as opposed to synthetic hair the primary specular highlight is shifted slightly more towards the root.

Light coloured hairs such as blond, brown, gray and white look very bright when lit from the back. This is due to light passing through these hairs which are more transparent then darker hairs. Marschener proposes a ray-tracing method to accurately simulate these and other properties but adaptations to the current shading model and compositing methods can also recreate them quicker and with more artistic control. Such methods are discussed in Section 6.1.

Chapter 3

The Pipeline

3.1 Software Used

When RenderMan is referred to in this documents it refers to *Pixar's RenderMan Pro Server 13.0.2* which is a RenderMan compliant renderer. The reason the term RenderMan is used is because many of the features of the renderer that are used are contained in the RenderMan technical specification[1] and hence are available in other renderers that are fully compliant with the RenderMan specification. The author will endeavour to point out features used that are not in the RenderMan specifications. The reason for using RenderMan is as follows:

- *It is widely used in the visual effects industry*

  Having been used on ground breaking films such as Stuart Little[SL199] and King Kong[Kon05] to name but a few, RenderMan has proven to be able to handle rendering of complex scenes and hair in a production environments.

- *Complete control over how the render happens*

  Every aspect of how one wants a scene to be rendered can be controlled which is a great benefit when one is trying out new techniques of rendering. Also unlike other renderers ray tracing and global illumination features are optional and turned off by default in RenderMan which primarily uses a form of Reyes scan-line rendering architecture[CCC87]. This is very useful when one is trying to render complex geometry such as hair.

- *Renderer independence*

  As mentioned above, RenderMan uses the RenderMan Interface Specification and therefore other renderers that are RenderMan compliant can render the same scenes and use the same shaders created for this project.

[1] See http://renderman.pixar.com/products/rispec/rispec_pdf/RISpec3_2.pdf for the entire specification

Sidefx's Houdini Master version 8.2.13, which has been and will continue to be referred to as Houdini in this document, was used to model the fur and other tasks such as placing camera and lights in this project. Reasons for using Houdini are:

- *Previous work*

  The system explained in this document was built on a previous system created described in [Kea07] which used Houdini.

- *Compatibility with RenderMan*

  Houdini's integration with RenderMan is superior to any other 3d package currently available. Its ability to render to RenderMan is built in to the core package as opposed to other 3d packages such as Autodesk's Maya which require a plugin to render to RenderMan. This compatibility also makes attaching RenderMan shaders to objects and creating batch renders extremely easy without the need to write a bespoke systems.

- *Used in the production*

  Houdini has been used to create visual effect features for many projects such as Superman Returns, Spider-Man 3, Monster's House and many more[2]. *Framestore CFC* even use it for their fur system for projects such as as the "Go Wild" project for the Rexona deodorant brand[Boy07]. This proves that Houdini is a production ready tool and is capable of the task.

- *It is a procedural package*

  Houdini was originally designed for procedural modeling and animation. Hair modeling is a procedural method. Also its Operator Type Library (OTL) system makes plugin creation extremely simply which makes it perfect for fast prototyping.

- *Attribute Transfer System*

  Houdini has a long history of being used for particle simulations and this maybe why it has such an advanced attribute transfer system. This system allows one to create, change and transfer attributes of a geometry efficiently and easily. For hair this means attributes such as hair length can be easily added to geometry and then transferred to curves to define the actual Length with ease and efficiency.

3.2 Pipeline Overview

Adding hair to a model, which is normally an animated creature, is usually one of the last parts of a linear computer animation pipeline. The pipeline used for this project, which can be seen in figure 3.1, takes place after the animation has been complete and the only thing left is to composite the images together

[2]See www.sidefx.com `http://www.sidefx.com` for a more extensive list

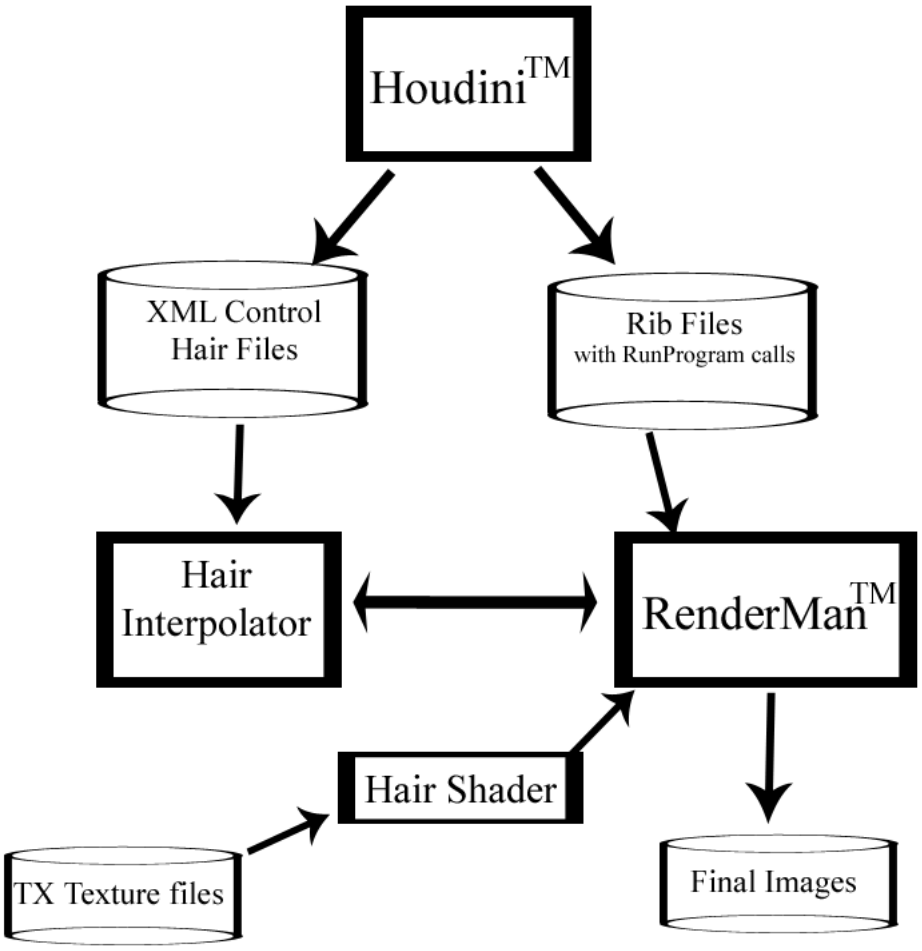

Figure 3.1: A diagram showing the pipeline used in for this project. Cylinders represent files and boxes represent programs

in a compositing package such as Shake. The animations for this project were loaded in as a series of obj geometry files. Due to the ubiquitous nature of the obj file format, the animation can be created by many applications such XSI's SoftImage which was used to create the polygonal model and animations for the rat model used for most of the examples in this project. The modeling of the fur is done on the "t-pose" of the model and hence can be done once the creature model has been created and the uv co-ordinates mapped to the model. This means it can be done parallel with the rigging and animation of the character which is extremely useful in a large production.

After loading in the creature model and modeling the hair, which is discussed in more detail in Chapter 4, Houdini creates an XML file which contains all the information needed by the fur interpolation program to interpolate the fur across the surface. The fur interpolation program which is named 'gerrycurl' is discussed in more detail in Chapter 5. A rib file is then created by Houdini which ordinarily contains the following:

- *The underlying skin or model as a matte object*

  There is an option in Renderman to create a 3d holdout matte (see page 76 of [AG99]). This matte object does not render in the final image and removes all the objects behind it. The rendering in this project is done in layers. The use of a matte skin makes it easier to composite the skin and the hair which is discussed in greater detail in Section 6.5.

- *A call to "RiProcedural" of type "Run Program."*

  RenderMan has a an riprocedural call of type run program which executes a program that emits RIB commands on its standard output stream at render time. An ASCII data block or a command line argument can be passed to this program. The only draw back is that a bounding box must also be declared that will be big enough to contain all the geometry created. Making this box too big is efficient but making it too small could lead to some geometry being clipped. In this project the 'gerrycurl' program is called and the XML file containing information about the control hairs is passed to it. The bounding box is calculated by Houdini based on the control hairs. See Chapter 5 for more information.

- *Guard hairs and other sparse hairs such as eyelashes*

  As mentioned in Section 2, fur contains sparse long course hairs called guard hairs which are similar to eyelashes on a human. Since these hairs are sparse and long it is simpler and more efficient to put these directly into the rib file from Houdini bypassing the interpolation system.

- *Regular information such as camera position, lights and surface shaders.*

  This information exists in nearly every rib. See [AG99] for more about creating a scene.

Chapter 4

Hair Modeling

4.1 RiCurves

The curve primitive also known as ricurve has existed in PRMan since version 3.7[app]. This primitive was used exclusively in this project to represent individual hair strands. This is because hair strands appear as very thin curves to the naked eye and ricurves are extremely efficient to render which is an important requirement when hundreds of thousands of them must be rendered. RenderMan renders the curves as ribbons defined by a curve called the spine. The ribbon is always orientated towards the camera which entails rendering less polygons then with a true generalized cylinder. The curve extends along the v direction. This property (which is useful during shading) means that v at the root or start of the curve is zero and at the tip or end is 1. The specific curve used in this project is a cubic non-periodic curve with four control points. Non-periodic means the curve does not wrap around in the v direction. Four is the minimal number of control vertexes for a cubic curve but is sufficient for short hair. The first and last point lie on the curve which is useful for modeling.

Two width values of type varying float are attached to the curve, one for the tip width and one for the root width. This width along the curve is defined by the user and is defined in object space. It is useful to have a higher width value for the root then for the tip. Even though real hair does not usually taper to a point like this it does make the hair appear more plentiful and in experiments were a constant width is used the hair appeared more synthetic. Increasing the width value and decreasing the number of hairs is one of the simplest controls for level of detail. Level of detail is an area of computer graphics that allows objects that are far away from the camera to be rendered in lower quality and hence faster but still appear to be as detailed as those objects that are closer to the camera. More work would be needed to implement a full level of detail system into this project, for example, to avoid popping when going from one level of detail set of values to another. [CHPR07] explains a more advanced method that uses this simple idea and also see [JC00] for more details on level

of detail. Similarly two colour values are passed to the renderer which can be overridden by the shader by using a texture to define the colours and passing this directly to the shader. Normals for the hairs are calculated by the shader as explained in Section 6.2. For more information on RenderMan curves see pages 84-85 of [Pix05] and pages 123-125 of [AG99].

4.2 The OTL interface

*Note:*This section uses some terminology unique to Houdini. The author has tried to make it accessible to non-Houdini users but access to the Houdini user guide would be helpful. Also see [WC06] for a good introduction to Houdini.

To create these properties and generate control hairs a Operator Type Library, referred to as an otl, was created in Houdini called "Gerrycurl." After installing the otl and loading in the animation and their model in a t-pose, the user is ready to add hair to it. First they select the animated model and rest model they wish to put hair on. The rest model is the model in a t-pose or equivalent. This is required to calculate the area of the model for scattering the model. The rest model also must contain normals that are not combed. The direction of the normals of the animated model can be used to define the direction of the hairs since Houdini contains a comb node that supplies a good user interface for defining hair direction. The normals of the underlying surface are still required for the shader so this why they must still be contained in the rest model. Also the direction of the normals can be saved and read from most 3d file formats meaning that the normals could be "combed" in an other application but this was not researched.

Once the model is correctly loaded in then the hair attributes can be defined. The visualization of the hair in the viewport must give the user visual feedback of what the hair will finally look like but also give immediate results. To control how approximate this visualization is, there is the visual speed parameter. This value is actually the number of hairs that appear on the model in the viewport and value assigned to it should be related to the speed of the end-users computer system. An example of this can be seen in figure 4.1.

The length of the hair is set by the maximum length parameter. As stated in [Boy07], for hair to look natural some noise or randomness needs to be added to it. In lieu of this there is an option to make the length random. This pseudo randomness is controlled by the randomness frequency parameter which is a value between zero and one. If set to zero then all the hairs will be the maximum hair length. The formula used is:

$$L_{max} - (f_r(p+S) * R_{freq} * L_{max})$$

$f_r(p+S)$ is a pseudo random number generator which returns a real number between zero and one inclusive. Houdini's rand expression is used to calculate this. $p+S$ is a seed number and p is a point id. The point id is an arbitrary

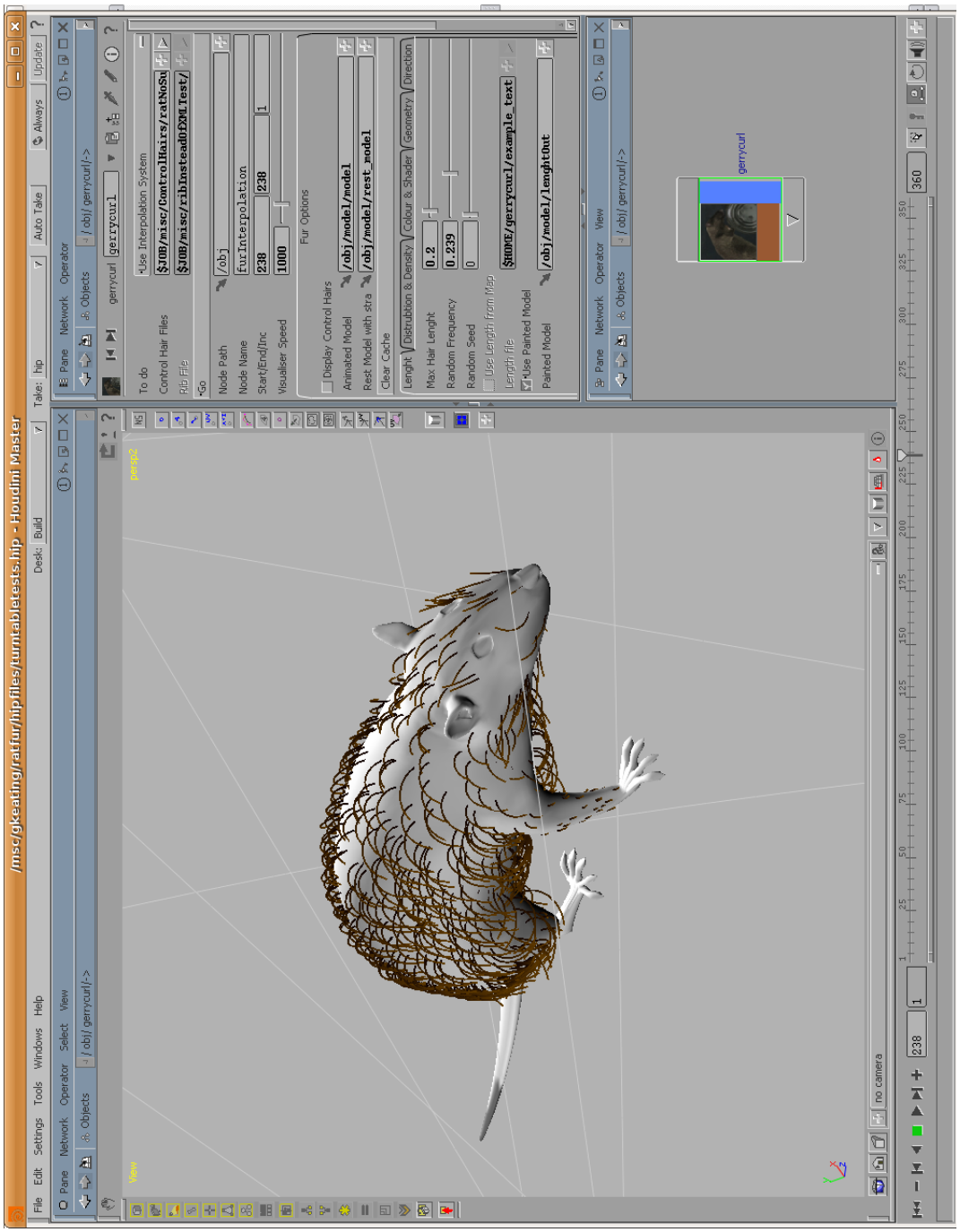

Figure 4.1: A screen shot of the fur modeling system being used in Houdini.

unique number created by Houdini for every point that does not change between frames even when the model is being deformed but it can be changed by a sort SOP. S is a seed set by the user. This value can be useful if, for example, one is creating two characters from the same model and they want them to appear different then they can simply set a different value for S for each character's hair. $L_{max}$is the maximum Length set by the user. R_{freq} is the random frequency set by the user. The maximum hair length can also be defined by a texture. The black areas of the texture will contain hairs of zero length and white areas will have hair of maximum length defined above. Similarly the user can paint the black and white colours directly onto the model and select this node to define the length.

The amount of hair on the final model is controlled by the hair density parameter. If one is bypassing the interpolation system this figure will be exactly the number of hairs on the final rendered model. Otherwise it represents a density value. The amount of hair for a given triangle must be proportional to the area of the triangle otherwise complex parts of the model will have a disproportionate amount of hair on them since hair is created per triangle. Therefore the amount of hair for a given triangle is calculated as a result of the hair density for that triangle and its area. This is why a rest model is defined. When a character is animated its topology will not change but the area of the individual polygons on its surface will. This change in area will lead to hairs being created or desttoyed between frames. Visually this will appear as popping of hairs. Like the length attribute, hair density can also be defined via a texture or paint node.

Root and tip colours are defined in same way i.e. they can be constant, defined by a texture or painted on. Root and tip widths of the hair are constant values. These values can be overridden by passing a texture to the shader. Noise is added to the colour by the shader at render time as explained in Section 6.4. To shape the hairs, Houdini's twist node is used. The twist node in Houdini is used to deform geometry in a number of different ways. The deformation used for the gerrycurl system is the bending type. To put this in context, a hair is initially a Bezier line. It is then deformed and transformed in its own object space before being copied to the surface of the model. The bending is controlled by bending strength which is self-explanatory and a bending roll off. Bending roll off is described as the attenuation of the deformation by the Houdini documentation. For the bending deformation it seems to rotate the curve around the pivot of the deformation. The exact algorithm that Houdini uses to do this deformation is unknown. The bending strength can be made random in the same way as length. Once the hair is bent it can be orientated that is to say rotated around its origin in object space. It can also be twisted adding a form of curliness. The twist parameter also has an attenuation factor called roll off and these parameters can be randomized.

Once the user is happy with the general look of the hair he then creates the node that will be used to render the hair. First he decides where the node should go in Houdini. A geometry node is created which must go into an obj level. The user then decides the name of the new node. For convenience the

user can also set the surface shader used even though this can also be set once the node is created. If generating a rib file the name and location of the rib file is set. Similarly for interpolation system the control hairs' file names and location is set.

4.3 The Houdini Network

Figure 4.2 shows the Houdini network used in this project. The animated model and the rest model described in Section 4.2 is loaded in. The area is calculated from the rest model. The surface normals and area are transferred to the animated model. The density attribute is then added and the UV coordinates of the surface are saved. The scatter function is now applied. This scatters points across the surface of the geometry. The scattering is based on the area multiplied by the density. In this network two scatter nodes are used. One for the viewport display and one used for when the rib is generated. The colour, orientation, bending strength and widths are then copied. A copy node is used to copy hair curves to every point. Bending, twisting and transformations are done to the curve based on values "stamped" from the points. The surface normal attribute is remapped to the rib format "uniform normal". Surface u and surface v are remapped to uniform floats.

A similar network is used for creating the control files but it does not contain remapping for the rib file nor does it contain the scatter nodes. The network does contain a triangulate node which triangulates the polygonal mesh. This is needed since the interpolation described in Chapter 5 is baed on trianlges.

4.4 Issues with the Previous Modeling System

As mentioned in Section3.1, this modeling system is based on the system created in [Kea07]. One of the design decisions made in [Kea07] was that the otl should encapsulate the entire modeling and rendering process of hair creation. The reasoning behind this were sound but while using this system it became unworkable and the decision was ultimately a wrong one. Houdini is a very powerful tool with many useful features. This became more apparent as the project progressed. Encapsulating fur creation and rendering meant many features were unusable without editing the OTL. Another problem with the encapsulation method is that every conceivable option that an end user would want must be included in the parameters window. This means a lot of unnecessary development and an unwieldy user-interface or restricting the users ability. A more modular approach using Houdini's own user interface paradigms means that the system is a lot more flexible and adaptable. The adding of new features is quicker, more efficient and less prone to bugs and errors. One of the first and most noticeable changes that the overturning of this decision has been is the separation of rendering and the modeling system. Before the rendering

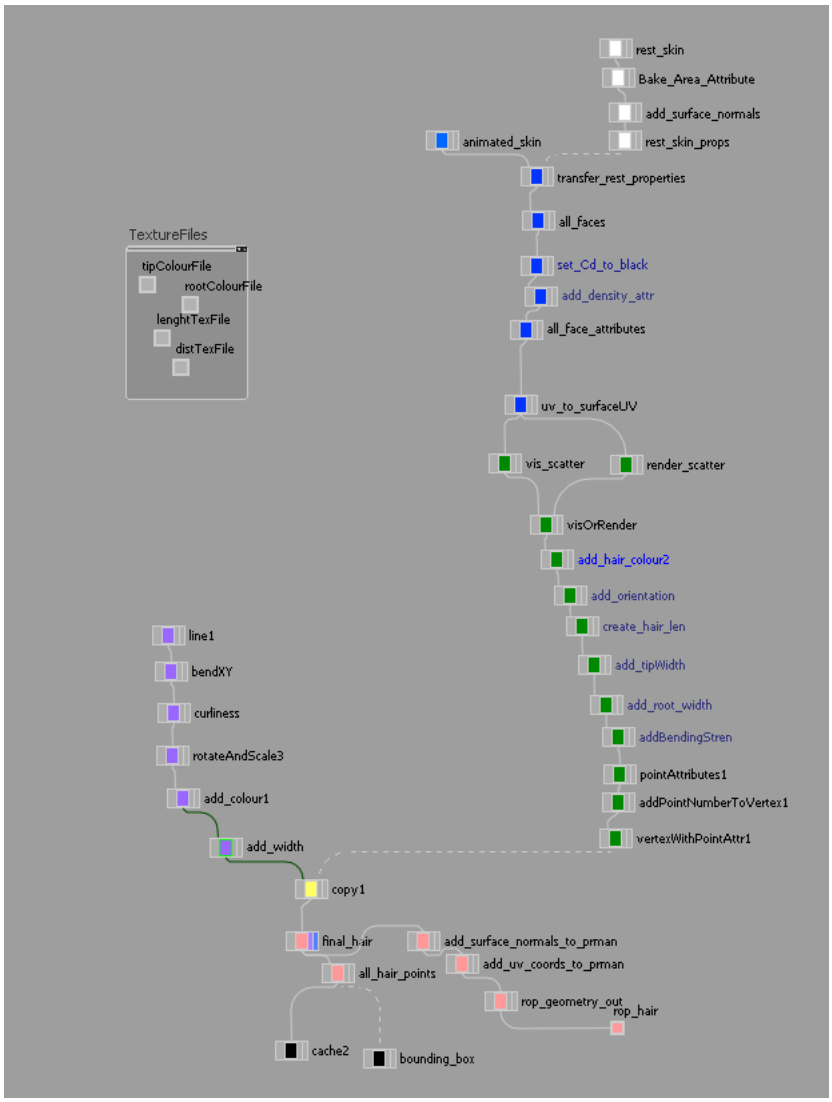

Figure 4.2: A screen shot of the Houdini node network used for visualizing the hair in the viewport and rendering the hairs directly to a rib file

happened within the otl. Now the otl creates a separate geometry node that contains a procedural call to the gerrycurl program or uses read archive to open a rib file. This conveniently allows a number of rman output nodes to render the fur, for example a separate node for every light could be created as was done in the example files. It also works better when working within a group. In many production environments the modeler will not light and render a shot. This new system allows the modeler to pass his work onto a lighting technical director with greater ease. Another noticeable difference is the removing of the guard hairs tab. Before, internally the features of the interpolated hair had to be copied to the guards hairs and changed slightly. This became tedious and error prone work. If the user wishes to create separate guard hairs that bypass the interpolation system then they can just create another node and change the node purpose option to create a rib file. Taking this modular approach to its extreme means that no otl should exist so obviously some compromise has to be made.

4.5 Future Work

There is a number of features that the modeling system is lacking. One of which is the clumping option. As noted and implemented in [Bre00] hair can have a tendency to clump together especially when wet.

Individual hair collision is not very important for hair as it is nearly impossible tell the difference between two hairs that are intersecting or merely touching. It could be an issue for dynamic hair but this depends on what kind of dynamics system is used. Hair colliding with the underlying geometry is an issue since this can lead to a popping affect as hairs disappear and reappear around the moving geometry such as at a shoulder joint. Combing and other modeling techniques could be used to lessen the issue. A better solution would be to use a dynamic collision detection system.

Houdini does have a fur node which was a prototype in version 8.2 and Houdini version 9, which was released during the completion of this project, has a full fur creation system. The features and usage of this new version is an area of further research. Comparisons and contrasts with other fur modeling systems such as Joe Alter Inc.'s Maya plugin, "Shave and Haircut[1]" or Worley's Sasquatch system[2] would also be an interesting area for research.

[1]See http://www.joealter.com/ `http://www.joealter.com/` for more information about this system. The website was last accessed in September 2007

[2]See http://www.worley.com/ `http://www.worley.com/` . Last accessed in September 2007

Chapter 5

Interpolation System Gerrycurl

5.1 Explanation

A hair interpolation system is defined, for this project, as a program that creates hair geometry at render time based on user defined control hairs as opposed to it being created by a program or user and then passed to the render to be rendered. There is many reasons for using such a procedural system. Firstly there is the memory consideration. Hair saved directly to a rib file can easily reach 100 or more megabytes in size. This may not seem like much but when one considers that this is per frame and there can be as many as 25-30 frames per second of the final film.[1] Therefore over 2 gigabytes are needed for a single second and this is before other elements of the scene are included. This does not only waste memory resources but also slows down the pipeline. As pointed out by [HS01] generating a file of that magnitude takes time as does reading a file of that magnitude. The problem is exasperated by the fact that reading and writing to disk is one of the most expensive processes in modern computing.

Another reason that an interpolation system is used is because a modeler cannot hand animate 300,000 individual curves [2]. Therefore a procedural method must be used at some stage during the hair creation and it is more efficient to do this at render time then any other time during the pipeline.

5.2 Control Hair Description File

The hair is interpolated based on user data files. In this project the data files are XML data files. XML (Extensible Mark-up Language) is a general-purpose

[1] This project use the PAL format which uses 25 fps as opposed to the American system, NTSC which uses 30 fps.

[2] 300,000 is the number used for a number of example renders in this project.

specification for creating custom mark-up languages. XML was developed by the World Wide Web Consortium[3]. Originally a bespoke file format was used that simply listed numbers. This became unwieldy as the project progressed since it was inflexible and difficult to extend. The XML format is useful because it is human readable which greatly aides debugging. It also has a number of free resources for efficient and simple file parsing and access. TinyXML[4] was used as the parsing tool for this project. XML also allows data to be stored in a tree like structure. This was an advantage for this project because each triangle has three control hairs associated with it.

Using XML can be inefficient for larger files since it adds a number of superfluous characters to the file. In our model control hairs and there values are copied numerous times since they can be shared by numerous triangles. Also the format is not natively supported by Houdini. The current method for saving the control information from Houdini is to use a hscript, which is Houdini's native scripting language similar to Mel and Autodesk's Maya. The ability to redirect the output of an echo command to a file is used to create the XML files. This method is extremely easy to implement but very slow. A better method would be to create a bespoke output driver for Houdini using its software development kit known as HDK. Writing a python script for Houdini version 9 is also an extremely easy yet efficient method for this but Houdini 9 was unavailable for the completion of this project. An alternative method is to write to a rib file. Houdini writes to a rib file very efficiently and one can add any properties to geometry in Houdini using the ROP geometry node and attribute node. Then one would need to use a rib parser such as the one written by Peter J Lewis[5] to read in the information by the interpolator. One issue with this approach is that the rib file does not have a built in tree structure and hence a completely different structure and evaluation method would need to be used. The information that is saved to the file is shown in table 5.1.

5.3 Algorithm and Implementation

To interpolate across a single triangle, first one must calculate the number of hairs that are required which is simply the hair density of the triangle multiplied by the rest area of the triangle. A Barycentric coordinate system is used to interpolate the hair across the given triangle which is based on the system used for NVIDIA's Nalu Demo which is documented in [PF05]. The system works by finding three values bA, bB and bC that add up to one. The algorithm is best explained by the following pseudo code shown on page 5.3.

These three values are then used to create the control hairs by weighting the values defined as geometry type point in table 5.1. For example if the

[3]See http://www.w3.org/ for more about the World Wide Web Consortium. The site was last accessed 7 September 2007

[4]See http://www.grinninglizard.com/tinyxml/ for more information. Site last accessed on the 7th September 2007

[5]see http://www.pjblewis.com/ for more about this software. Site last accessed on the 7th September 2007

Property Name	Geometry Type	Data Type	Description
Triangle ID	Triangle	integer	A unique ID for the surface of the triangle that does not change between frames. Used for seeding the random number generator.
Hair Density	Triangle	float	How sparse the hair should be for this triangle.
Rest Area	Triangle	float	The area of the triangle when it is in a t-pose or similar state. Used for calculating the number of hairs on the surface.
Surface Normals	Point	normal(float[4])	The surface normal at the base of the control hair. This used for shading.
Surface UV coordinates	Point	float[2]	The UV coordinates of the surface at the control hair base.
Root Width	Point	float	The width of the hair at its base.
Tip Width	Point	float	The width of the hair at its tip.
Root Colour	Point	colour(float[3])	The colour of the hair at its root which can be overridden by the shader.
Tip Colour	Point	colour(float[3])	The colour of the hair at its tip which can be overridden by the shader.
Control Point Positions(4)	Point	3d point (float[3])	The main purpose of the file. The positions of the four control points that will create the hair.

Table 5.1: A table describing the information saved to the control hair file. The columns describe the name of the data that is being saved, whether it is saved per triangle or point, the data type and a description of it respectively. As can be seen, the data is split into two types. Those associated with the triangle and those associated with control hairs referred to as "Point" in the table. The triangle data is used to calculate the number of hairs that will exist on the triangle and its id. The rest of the data is used for the hair properties.

```
bA=getRandNumber(seed);
bB=getRandNumber(seed);
sum=bA+bB;
if(sum>1) then
  if(bA>bB) then
    bA=1-bA;
  else
    bB=1-bB;
  end if
  sum=bB+bA;
end if
bC=1-sum;
```

base points of the control hairs are $P1$, $P2$ and $P3$ respectively then the newly created control hair's point is calculated by:

$$P1 * bA + P2 * bB + P3 * bC$$

All the other values for the new hair are calculated in the same way. The newly created hair is then passed directly to the renderer and overwritten by the next hair. The implementation of this system is done in C++.

Figure 5.1 is a class diagram of the code used. Main reads in the control hair XML data file name either from the command line or from the standard input stream. The file is opened and then for every triangle encountered a triangle class is created. Triangle repeats the algorithm from page 22 for every hair on its surface to be created. Triangle then creates a hair object that dumps its values onto the standard output stream in the rib format. No more then one triangle object or hair object exists in memory at one time since they are overwritten in every loop. The advantages of this interpolation system is that it is extremely fast. Calling a random number is the most expensive procedure and this is only done twice for every hair created. Speed is a very important factor since this procedure will be called hundreds of thousands of times per frame. The interpolation works in two modes. The command line version is called as such:

gerrycurl f controlHairsFile.xml

Were controlHairsFile.xml is the name of the XML file that contains the file. The program then precedes to print out the resulting rib code to the standard output. This is extremely useful for debugging the code. It also is useful if one wishes to cache the rib information. The rib file can be read in using RenderMan's read archive function. Caching the rib information in this way can be extremely useful for a number of shots were the hair does not move and the underlying skin does not deform in anyway. An example of when this occurs is short hair on a human head. This method was used extensively in Final Fantasy: Spirits Within [Bjo01]for the male characters. The main purpose for gerrycurl is for it to be run at render time. The following example from a rib file shows its use:

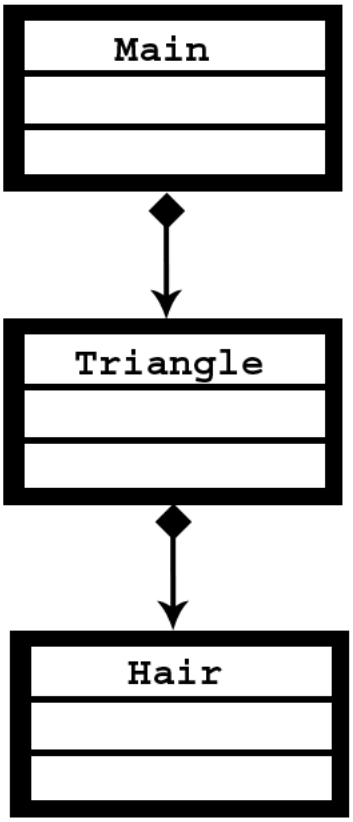

Figure 5.1: An UML class diagram demonstrating the fur interpolation system.

Procedural" RunProgram" ["*gerryCurl*" *controlHairsFile.xml*"] [0 1 0 1 0 1]

The list of the numbers at the end are the bounding box. This is required by RenderMan to set aside a 3d volume during the clipping phase of its rendering process. Hence a bounding box that is too big is very inefficient but to have one too small leads to parts of the newly created geometry being clipped from view. For this project the bounding box was created by Houdini and is based on the control hairs.

5.4 Limitations and Future Work

A number of optimizations could be added to this system. One of the most glaring ones is that it writes to the standard output. A more efficient method would be to make it a DSO (dynamic shared object). This means that the renderer will call it once the subdivision routine is called and then it will be called as if it is statically linked. This removes the overhead of interprocess communication. See page 120-121 of [AG99] and [HS01] for more in depth information regarding DSOs.

The interpolation system works on a flat polygonal surface. For the Nalu demo this is not noticed since the hair is long and dynamic but for hair combed tight to the surface of the object the model can appear faceted. That is one can see the underlying simplicity of the polygonal model. A solution to this issue would be to use Non-Uniform Rectilinear B-Splines (NURBS) models and interpolating across a NURBS patch rather then a triangle. This requires your model to be a NURBS model or converted to a NURBS model which many fur systems require. This solution was used for Stuart Little[Bre00].

Randomness is an important part of fur [Boy07]

Randomness can be added to nearly every attribute of the hair. For example in the modeling system the artist can add randomness to the bending strength (see Section 4.2) but there is an issue with this and the current interpolation system. Lets suppose a hair is randomly bent while surrounding control hairs are straight. The interpolated hairs will be linearly interpolated from straight to bent as demonstrated in figure 5.2.

This may not be what the artist wants. To work around this one could make the control hairs change very gradually and have another set of hairs added to the rib file that are completely random and bypass the interpolation the system. Another solution would be that instead of interpolating control points one could interpolate the properties used to generate the hairs including the randomness properties. The hairs would then be deformed and transformed at render time. Basically doing what Houdini does to create the control hairs but doing it at render time. This would mean more work needs to be done at render time by the interpolation system but it would lead to more reliable results for the artist.

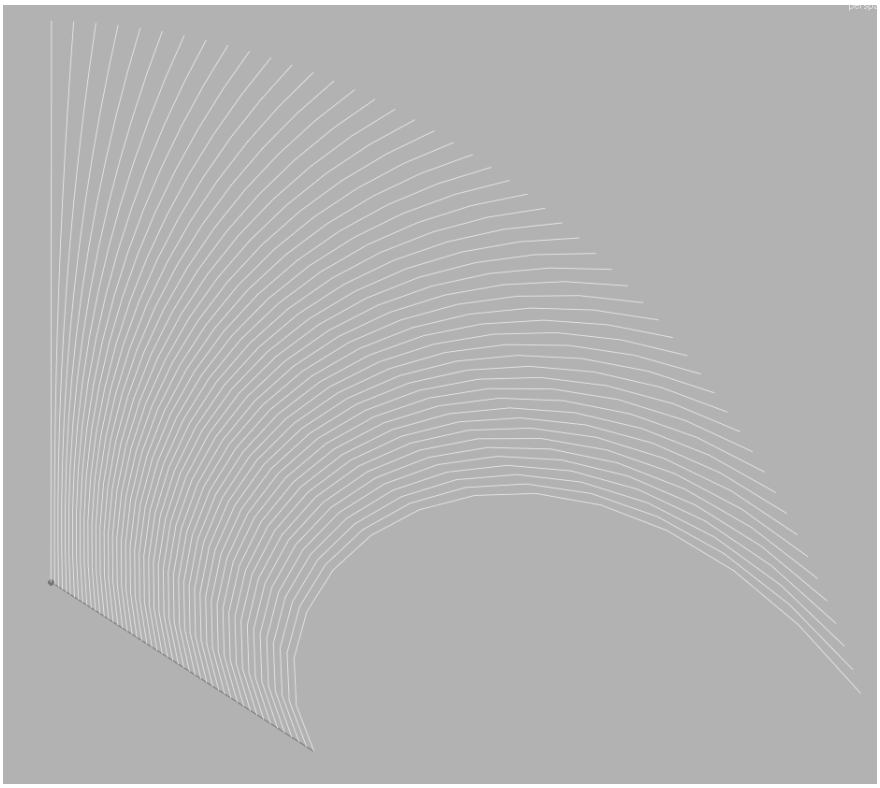

Figure 5.2: Demonstrating how, in one dimension, one hair's random change in direction can change surrounding hairs. The hairs at both ends of the line are control hairs. This problem is exasperated in the actual system since a control hairs are shared by any number of triangles.

Chapter 6

Shading and Composting

6.1 Kajiya Kay Shading Model

In the 1989 landmark paper [KK89] James T. Kajiya and Timothy L. Kay proposed a method for rendering furry objects. There method extended work done in [Bli82] and [KH84]. The majority of the paper describes a method for ray-tracing hair but what has made it famous is the method it uses for calculating the specular highlight of a strand of hair. The model is based on the Phong model and has become known as the 'Kajiya, Kay model'.

The hair is assumed to be cylindrical. The specular light is reflected at a mirror angle to the tangent of the hair. Assuming the cylinder has normals pointing in all directions perpendicular to the tangent, this reflected light creates a cone as shown in figure 6.1 . The actual formula for the highlight intensity is:

$$\Psi_s = k_s \cos^p(e, e\prime)$$

ks is the specular co-efficient, e is the vector pointing to the eye, $e\prime$ is specular reflection vector closest to the eye and p is the Phong exponent. The formula is basically the cosine of the angle between the specular reflection and the eye vector raised to the power of Phong exponent. The Phong exponent is in many specular models. It is a measure of the smoothness of the surface or put another way, it is the inverse of the roughness of the surface. Many shaders use the latter definition.

If we say that θ is the angle e makes with the hair and $\theta\prime$ is the angle that $e\prime$ makes with the hair then we can rewrite the formula as:

$$\Psi_s = k_s \cos^p(p)(\theta - \theta\prime)$$

$$= k_s(\cos\theta\cos\theta\prime + \sin\theta\sin\theta\prime)$$

$$= k_s((t \cdot l\, t \cdot e) + \sin(t, l)\sin(t, e)^p)$$

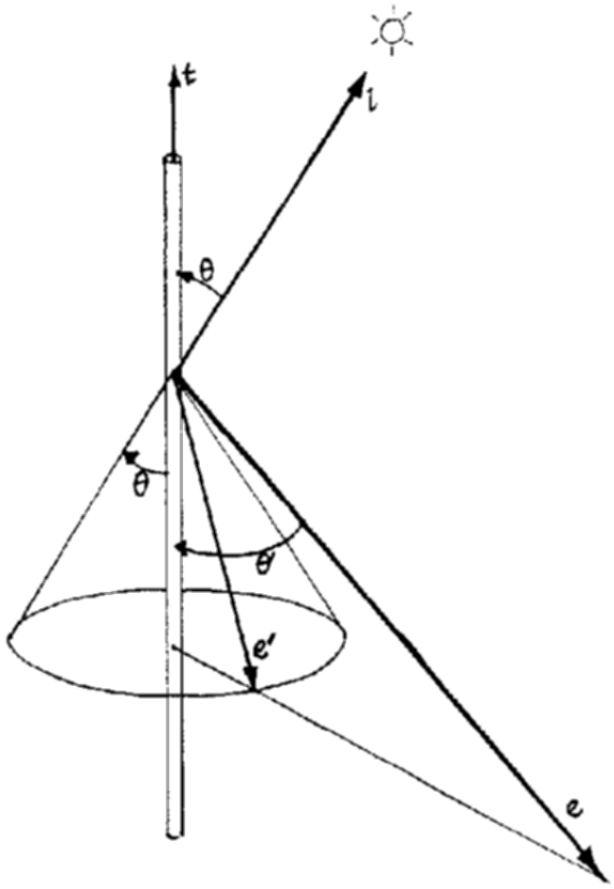

Figure 6.1: A diagram taken from [KH84] showing a hair as a cylinder and the specularity cone used.

From line 1 to line 2 is a simple trigonometric definition. t is the tangent of the hair and l is the light vector. If the two vectors are normalized then the dot product of the two vectors is equal to the cosine of the angle between them. This fact is used to derive line 3.

The "Kajiya, Kay" specular model is extremely useful. It has been used extensively because it is extremely fast, it has even been used in real time applications[Sch04]. It is simple to implement in a number of different rendering models such as scan line or ray-tracing system. The model can also be adapted. One adaptation used in [Sch04] is to move the tangent vector along the normal to the hair. Using the formula:

$$T_{new} = T + S * N$$

S is a scalar value, which is the shifting amount. This can be used to break up the specularity of the hair so it appears more random and allows the artist more control of the specular highlights. The normal in this case is simply perpendicular to the hair tangent. A dual highlight can be added by having two different values for S.

6.2 Hair Normals

The diffuse component of the hair for this project is found by calculating the dot product of the normalized light vector and the normal of the hair. Kajiya-Kay had a similar method for the diffuse component which was $\sin(t, l)$. Depending on the relation between t and n, the two methods could be equivalent.

The method for working out the normal of the diffuse component is based on method used in [Bre00]. One finds the cross product of the surface normal and the hair tangent then linearly interpolates between that value and the surface normal. The linear interpolation is weighted by the angle between the tangent of the hair and surface normal. The formula for this is as follows:

$$\alpha = S \cdot T \tag{6.1}$$

$$N = \alpha S + (1 - \alpha)(S \times T) \tag{6.2}$$

This means that if the hair is parallel to the surface, its normal will be the same as the surface normal and if it is perpendicular the normal that is perpendicular to the hair is used. This makes the hair lighting much more intuitive for people used to lighting regular surfaces.

6.3 Deep Shadows

Shadows add depth and complexity to any model. They are particularly important for hair since hair has a significant amount of self shadowing. The regular

method for creating shadows involve either ray tracing or z-depth[RSC87]. For complex geometry, like hair, ray tracing is too computationally expensive for production. The z-depth method for RenderMan involves rendering a z file from the light source. That means using the light as the camera. The z file contains the distant from the light to the shading point for every pixel. At render time the distant from the camera is compared to the distance stored in the z file by transforming the light to the camera co-ordinate space. If the distances match, the point is not in in shadow but if the point is further away in the camera then in the z file then the point is in shadow. This is because something is occluding the point in the z-file. For a better explanation of this method and for implementation tips see page 235 to 241 of [AG99].

For accurate shadowing of hair the resolution of the z-file must be extremely big. This causes issues both for memory and also look up time. Deep shadows[LV00] alleviates these two problems by storing fractional visibility through each pixel at all possible depths or put another way, a visibility function for each depth. It is also uses pre-filtering which has faster look up times and uses less memory. From an end users point of view, deep shadows appear to be the same as z files but have better quality. One big difference is that with deep shadows they can have many samples per pixel.

The RenderMan Interface[Pix05] does not include any references to deep shadows but it is supported by PRMan and is relatively easy to implement so it is supported by many other renderers.

6.4 Colour

As was said on page 24 hair is defined by its randomness. Due to this, noise is added to the hair colour. Noise can be viewed as a method for generating pseudo-random numbers. For this project two types of 2d noise, which uses the u and v co-ordinates of the underlying surface, were used. One type is of high frequency which means the noise changes significantly from hair to hair and a low frequency noise which means that vast areas of hair have the same noise applied to them. The exact number for these two frequencies are user defined shader parameters. These noise functions create colour that is then mixed with hair colour defined by the user. The mix is weighted by a user defined shader parameters.

6.5 Implementation and Composting

Rendering in passes is essential in production rendering. It allows changes to be made to the final images interactively and is essential for shot integration(see pages 256 to 271 of [Bir00]). Rendering multiple passes was achieved using arbitrary output variables in the surface shader. Applying these allows one rib file and one shader to output any number of images. For this project a specular pass was outputted which outputted the result of the Kajiya, Kay specular

described in Section6.1. The diffuse pass which outputs the results explained in Section 6.2. The shadow pass which outputs the shadow information described in Section 6.3. The shadow pass is slightly counter intuitive. If a point is in shadow it has a value of one but if it is fully illuminated it has a value of zero. The colour pass which outputs the colour of the object and the alpha pass outputs the opacity of the object but these are superfluous since the default output contains these values.

Many of the shading parameters can be defined by maps as was used for Aslan's fur in Narnia described in [HDK+06].

More passes could trivially be added to the shader but this would require a lot more work in compositing. The images generated were rendered in the OpenExr format which allowed for full 32 bit floating point values. This is a great benefit in compositing since it allows more changes to be made without doing a re-render. Doing gamma correction in post rendering for example would mean losing information for most formats but no information is lost when using floating point.

The compositing was done using Shake which is a standard industry compositing package. The compositing tree can be seen in Figure 6.2.

As per standard the diffuse pass is multiplied by the colour. The specular is then added to this and the shadow pass is used as a mask for the brightness. There is various other alterations that can be done to the passes in the composition such as changing their brightness and colour.

6.6 Conclusion

The shading has worked extremely well with some exceptional results. More time could lead to more tweaking of the many parameters both of the shader and in composition. Some more work could be done to research how it can be integrated into a live action scene.

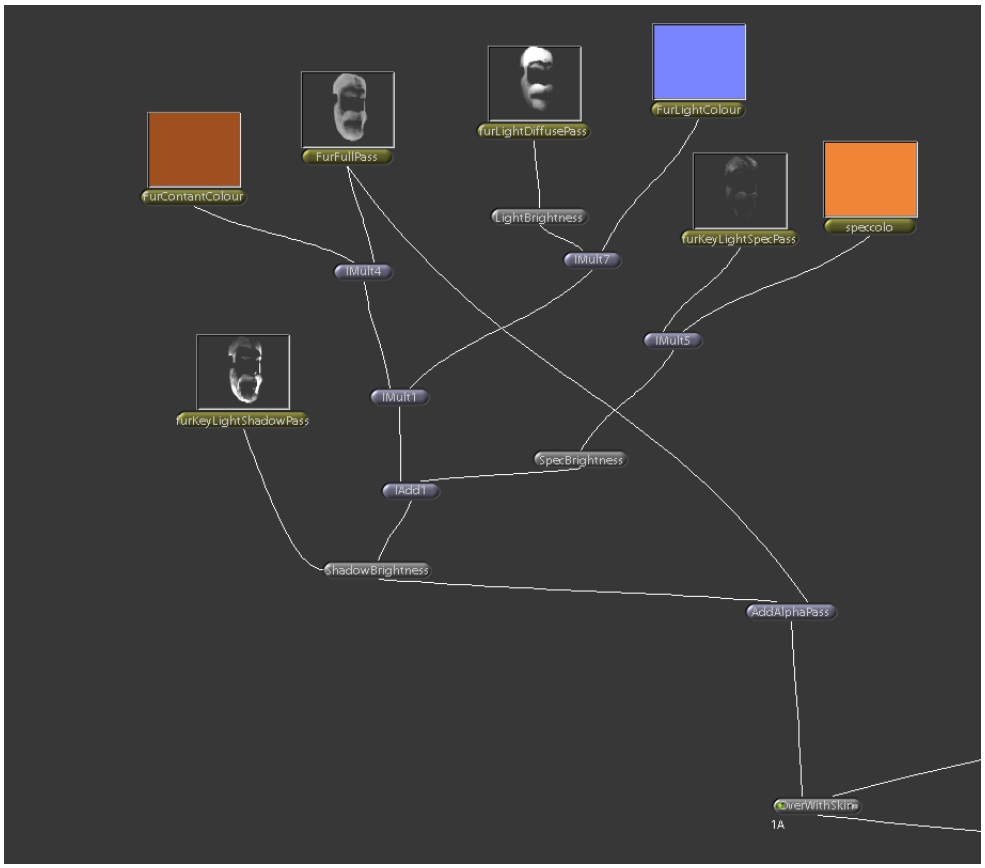

Figure 6.2: Part of the node view taken from Shake used for the creating the final composite. The final node shows this part of the network is before it is "overed" with the skin. This is just an example of how the passes could be composited. A more experienced compositor may know better methods.

Chapter 7

Conclusion

The creation and rendering of short hair and fur is a large area for research. Pipeline creation, modeling, interpolation systems, shading and rendering hair are all large areas that were only touched on in this project. Each of these areas could easily take up an entire master thesis but the goal of this project was to present an entire system.

The area that needs the most work is the interpolation system which has a number of deficiencies described in Section 5.4. This is a major bottleneck in the pipeline and an area with the most detrimental affect on renders. This could be remedied by any of the methods described in Chapter 5 such as a better Houdini control file exporter and interpolating across a NURBS patch.

Houdini has proved to be a good tool for fur modeling and it will be interesting to see how future versions of the software will advance the fur system.

It was said in the introduction that this project did not use dynamics system. Adding dynamics to the system presented here would be an interesting area for further research.

The results shown in Appendix A are reasonably good but the focus of the project has always been to give artists as much control over the creation of the final image as possible. Getting more artistic input and seeing the system being used for an actual production would be a true test of its value.

The project has created a full working system and using that criteria it has been a success.

Bibliography

[AG99] Anthony A. Apodaca and Larry Gritz. *Advanced RenderMan: Creating CGI for Motion Picture*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[app] Using the ricurves primitive. PhotoRealistic RenderMan Application Note #19.

[Bir00] Jeremy Birn. *Digital Lighting and Rendering*. New Riders Publishing, Thousand Oaks, CA, USA, 2000.

[Bjo01] Kevin Bjorke. Advanced renderman 3:render harder. pages 123–125, August 2001.

[Bli82] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *SIGGRAPH Comput. Graph.*, 16(3):21–29, 1982.

[Boy07] Andy Boyd. Fur real. *3d World*, (89), February 2007.

[Bre00] Rob Bredow. Advanced renderman 2:to ri infinity and beyond. pages 89–104, July 2000.

[CCC87] Robert L. Cook, Loren Carpenter, and Edwin Catmull. The reyes image rendering architecture. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 95–102, New York, NY, USA, 1987. ACM Press.

[CHP+79] C. Csuri, R. Hackathorn, R. Parent, W. Carlson, and M. Howard. Towards an interactive high visual complexity animation system. In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 289–299, New York, NY, USA, 1979. ACM Press.

[CHPR07] Robert L. Cook, John Halstead, Maxwell Planck, and David Ryu. Stochastic simplification of aggregate detail. *ACM Trans. Graph.*, 26(3):79, 2007.

[Cre92] Joan G. Creager. *Human Anatomy and Physiology Second Edition*. WCB, 1992.

[ff001] Final fantasy: The spirits within, 2001.

[HDK+06] Brad Hiebert, Jubin Dave, Tae-Yong Kim, Ivan Neulander, Hans Rijpkema, and Will Telford. The chronicles of narnia: the lion, the crowds and rhythm and hues. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 1, New York, NY, USA, 2006. ACM Press.

[HS01] Christophe Hery and Douglas Sutton. Advanced renderman 3:render harder. pages 73–96, August 2001.

[Inc04] The incredibles, 2004.

[JC00] M Reddy A Varshney B Watson J Cohen, D Luebke. Advanced issues in level of detail, 2000. Course Notes #41 of SIGGRAPH 2000.

[Kea07] Gerard Keating. A fur interpolation system using houdini and renderman. A Master's project created at the NCCA under the tuition of J Macey, June 2007.

[KH84] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA, 1984. ACM Press.

[KK89] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 271–280, New York, NY, USA, 1989. ACM Press.

[Kon05] King kong, 2005.

[LV00] Tom Lokovic and Eric Veach. Deep shadow maps. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 385–392, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[MJC+03] Stephen R. Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. Light scattering from human hair fibers. *ACM Trans. Graph.*, 22(3):780–791, 2003.

[MTHK02] Nadia Magnenat-Thalmann, Sunil Hadap, and Prem Kalra. State of the art in hair simulation. 2002.

[nar05] The chronicles of narnia: The lion, the witch and the wardrobe, 2005.

[OT87] J. P. Ortonne and J. Thivolet. Hair melanin and hair color. *Hair Research*, pages 146–162, 1987.

[PF05] Matt Pharr and Randima Fernando. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005.

[PH06] Martin Preston and Martin Hill. Grooming, animating & rendering fur for king kong. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 43, New York, NY, USA, 2006. ACM Press.

[Pix05] Pixar. The renderman interface version 3.2.1, November 2005.

[Rob99] James Robertson, editor. *Forensic Examination of Hair*. Taylor and Francis, 1999.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 283–291, New York, NY, USA, 1987. ACM Press.

[Sch04] Thorsten Scheuermann. Practical real-time hair rendering and shading. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, page 147, New York, NY, USA, 2004. ACM Press.

[SL199] Stuart little, 1999.

[SL202] Stuart little 2, 2002.

[WC06] Jason Iversen Dave Johnson Will Cunningham, Peter Bowmar. *The Magic of Houdini*. Thomson Course Technology, 2006.

Appendix A

Some Rendered Examples

Hair applied to the Cave Troll model created by Ritchie Moore.

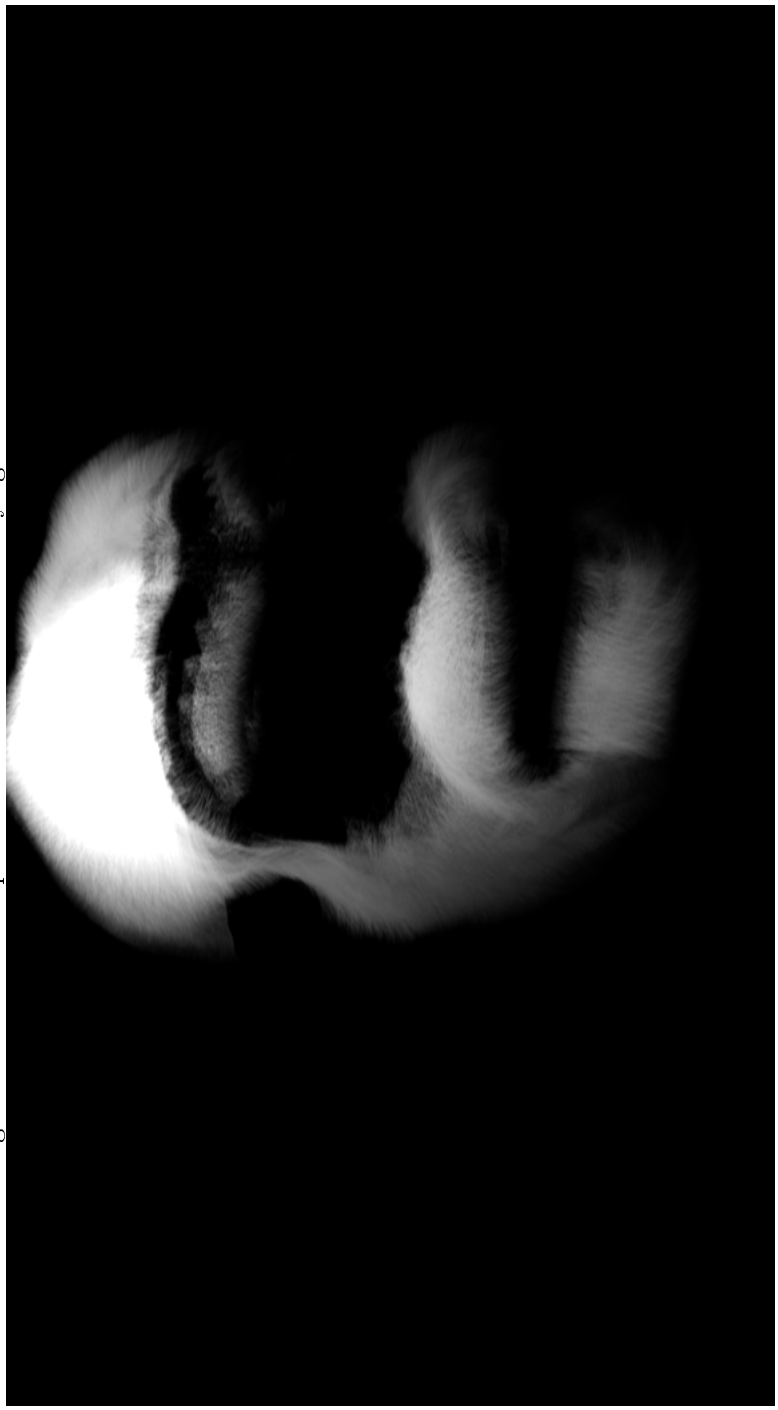

Figure A.1: The diffuse pass of the hair with the underlying skin as a holdout matte

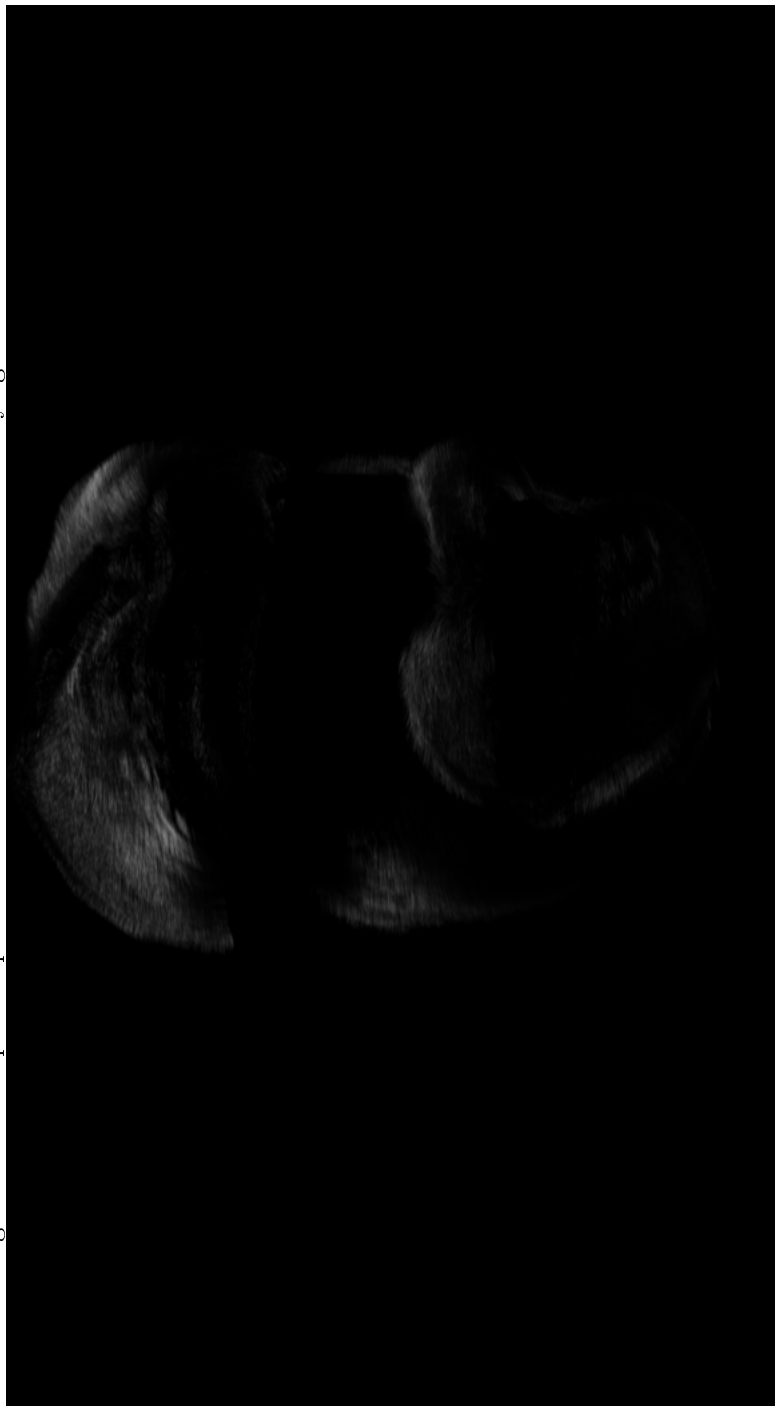

Figure A.2: The specular pass of the hair with no colour with the underlying skin as a holdout matte

Figure A.3: The shadow pass of the hair with the underlying skin as a holdout matte. The parts in shadow are in black and the parts in white are in shadow

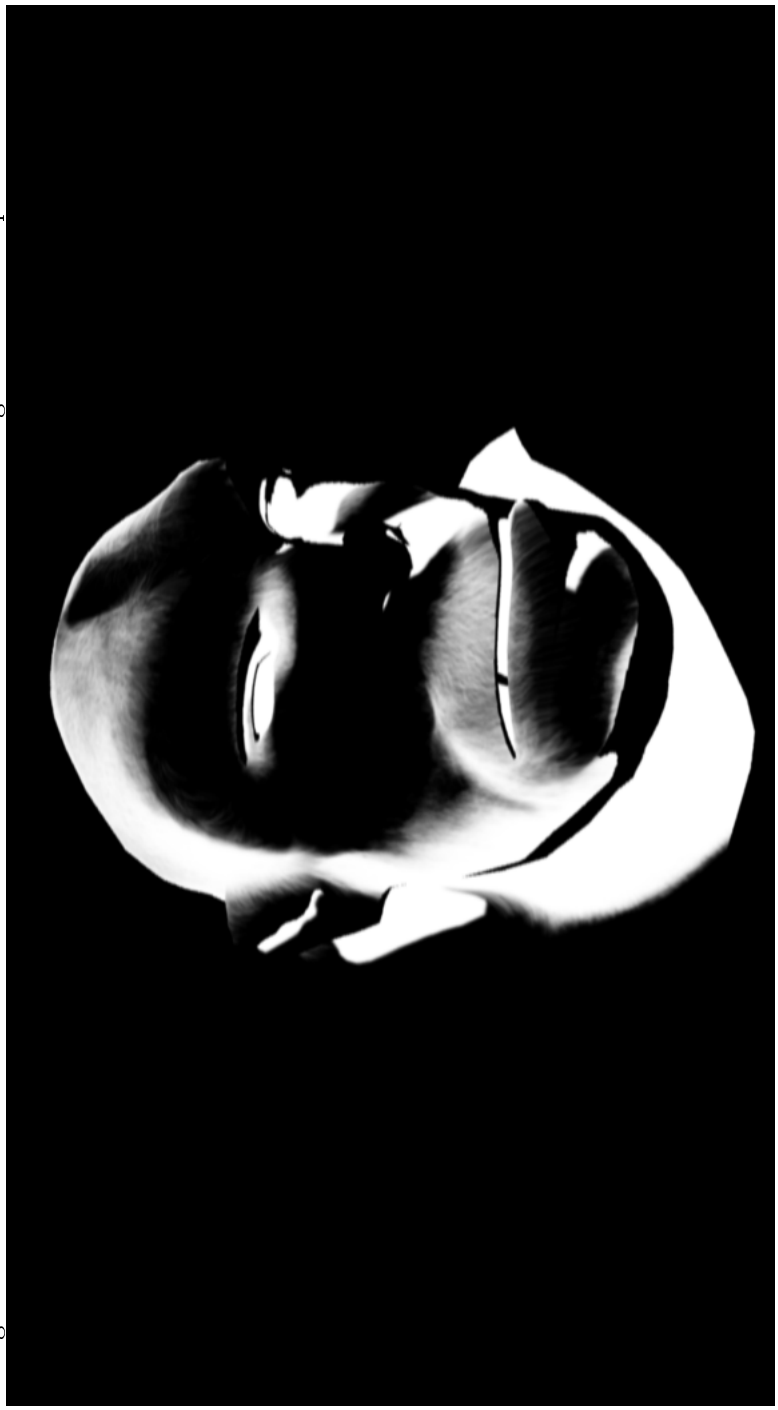

Figure A.4: The shadow casted onto the skin from the hair without the hair being visible. White parts are in shadow.

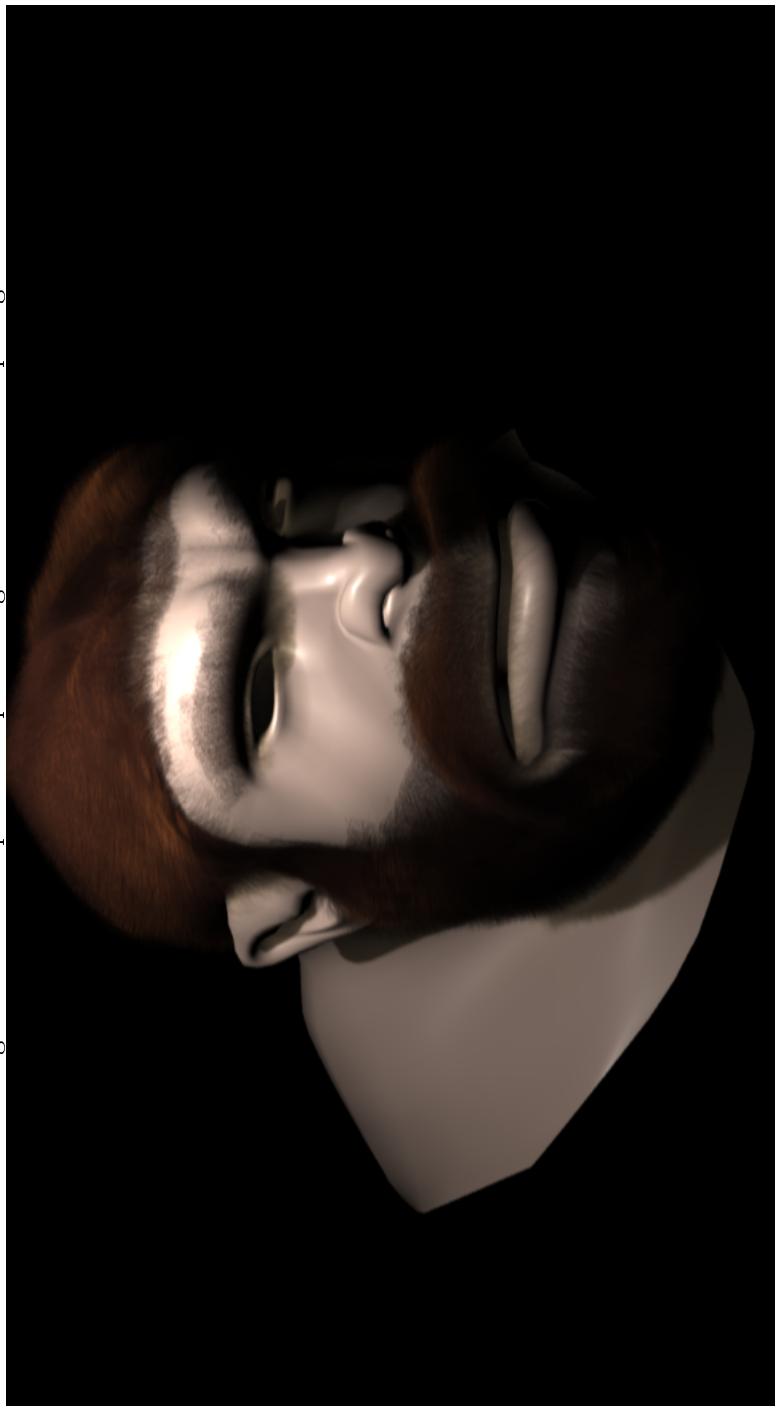

Figure A.5: All the passes composited together in the Shake package.