



A User Friendly Hair Reflectance Model in Houdini

Master of Science Computer Animation and Visual Effects
National Centre for Computer Animation
Bournemouth University

Daria Kozlova

Bournemouth, United Kingdom

August 2019

Acknowledgements

It would not be possible to completed this thesis without help from a great many people.

I would like to express gratitude to our advisors Jon Macey and Phil Spicer. They have been extremely supportive within all the year and have been always approachable whenever I needed any advice.

Furthermore, I would like to acknowledge my undergraduate advisor Vladimir Anatolievich Zakharov from Moscow State University. He have taught me well how to do scientific research.

Thanks to all MSc CAVE 18/19 students for a wonderful time – Tom Ashby, Ben Carey, Bianca Farquharson, James Preston, Clarence Rajaratnam, Jamie Slowgrove, Leah Sreshta, Rachel Strohkorb and Alec Watkins. They made this year unforgettable. Also, it is impossible not to mention my friends from another course, with whom we spent long time together on working on our group project – Tom Bailey, Zac Ives, Emre Sumer and Victoria Yu.

Last but not least I would like to thank my family for all the love and support.

Vita

2016 – B.S. in Applied Mathematics and Computer Science, Faculty of Computational Mathematics and Cybernetics, Moscow State University

2019 – M.Sc in Computer Animation and Visual Effects, National Centre for Computer Animation (NCCA), Bournemouth University

Abstract

High-quality realistic rendering of hair and fur is a long-standing goal in computer graphics. It includes a number of challenges, such as computation of accurate light scattering for individual fibres and handling the complex geometry of hair volume. Physically based approach in rendering hair is aims to achieve plausible result; however, its drawback is the lack of artist friendly controls.

In this paper we present an implementation of a user-friendly hair reflectance model in Houdini based on proposals in (Chiang et al., 2016) and (Yan et al., 2017), who constructed physically based hair and fur shading models. The results of these approaches are illustrated in numerous renderings. All images were rendered using the physically plausible pipeline in Mantra.

Contents

Acknowledgements	i
Vita	ii
Abstract	iii
1 Introduction	1
1.1 Motivation	2
2 Literature Review and Theoretical Background	4
2.1 Anatomy	4
2.2 Literature review	5
2.2.1 Physically Based Rendering	5
2.2.2 Hair Shading Models	5
2.2.3 Fur Shading Model	10
2.3 Houdini and VEX	13
2.4 Hair rendering in production renderers	14
3 Implementation	17
3.1 Pipeline	17
3.2 Shader Implementation	18
3.2.1 Extended Hair Shader	19
3.2.2 Fur Shader	21
3.3 Render Adjustment	22
3.4 Simulation	25
3.5 Render Passes and Lobes	26
3.6 Rendering and Assembling	26
4 Results	29
4.1 Extended Hair Shader	29
4.2 Fur Shader	30
5 Known Issues and Further Work	33
5.1 Further Work	33
5.2 Known Issues	34

6	Conclusions	36
	References	36
	Appendices	41
A	User Manuals	41

List of Figures

1.1	Usage of furry computer generated characters in one of the recent animated feature, (Peter Rabbit 2018)	1
1.2	Examples of fur, produced by DreamWorks Fur Motion System – Skunk, (Augello and Somasundaram, 2019)	1
1.3	Example of grooming in SideFX Houdini, (Browne, 2014)	2
2.1	Cuticle, cortex and medulla layering, (Morganti and Yuan Li, 2015)	4
2.2	From left to right: structure of human hair; cuticle layer on human hair fibre; example of fur fibre; cuticle layer on animal fur fibre, (Galatík et al., 2011), (Wei, 2006)	4
2.3	A scheme of hair model for one fibre from (Marschner et al., 2003)	7
2.4	Logistic (red) and wrapped Gaussian (blue) distributions, (Chiang et al., 2016)	9
2.5	All scattered (by medulla) paths in fur shading approach, (Yan et al., 2015)	11
2.6	Optimised fur lobes, (Yan et al., 2017)	12
2.7	Example of VOPs network in shader context in Houdini	13
2.8	Standard hair shading models in some production render engines. From left to right: Mantra, Arnold and RenderMan.	15
2.9	Left: R, TT and TRT lobes (equals to reflect, diffuse and refract) from standard hair shading model in Mantra. Right: direct (left) and indirect (right) diffuse and specular components of standard hair shading model in RenderMan.	15
2.10	3Delight Hair Model, (DNAResearch, 2013)	15
3.1	CyHairLoad Asset	17
3.2	Ambient rendering of <i>cyHair</i> examples. Models from (Yuksel, 2006)	18
3.3	Simple Nuke Script to run through nodes in Node Graph	18
3.4	Internal code in PBR Hair Shader Extended node.	19
3.5	PBR Hair Shader Extended parameters	19
3.6	PBR Fur Shader parameters	21
3.7	The plots of logarithm values of measured raw reflectance data for bobcat and cat, from (Yan et al., 2016).	21
3.8	From left to right: shift -1, 0 and 1.	22
3.9	From left to right: longitudinal roughness 0.1, 0.25 and 0.6. The hair with longitudinal roughness 0.1 looks shiny, while hair with longitudinal roughness 0.6 looks flat and diffuse.	23

3.10	From left to right: azimuthal roughness 0.1, 0.25, 0.5 and 0.7. The hair with azimuthal roughness 0.1 looks darker than hair with azimuthal roughness 0.7.	23
3.11	From left to right: alpha 2 and 4.5.	23
3.12	Example of various hair colours.	24
3.13	Medulla index 0 and 1 correspondingly. When medulla index equals 0, the main colour of the hair is taken from base colour parameter, and when medulla index equals 1 – from medulla colour respectively. In-between values are calculated in accordance to absorption function from Table 2.3.	24
3.14	From left to right: medulla scattering 0.01, 0.065 and 0.2.	24
3.15	From left to right: cuticle layers 0.1 and 2.45.	25
3.16	Left: renders with motion blur and without motion blur. Right: node network for producing hair simulation.	25
3.17	From left to right: R , TT and TT^s , TRT and TRT^s . Model from (Stavginski, 2017).	26
3.18	Work in progress Nuke script with assembling all renders.	26
3.19	Work in progress Houdini scene with various shader tests.	27
3.20	Work in progress Final Cut editing.	27
3.21	Directory structure for \LaTeX compiling.	28
4.1	Used HDRIs. From left to right: (Zaal, 2017b), (Mischok, 2019b), (Zaal, 2017a), (Mischok, 2019a) and (Zaal, 2016).	29
4.2	Three hair models with same hair shader under two different environment lights.	29
4.3	From left to right: low quality, middle quality and high quality.	30
4.4	The Cornell Box test for extended hair shader and example of separate render passes.	30
4.5	Comparison of default hair (left) and fur (right) shaders.	31
4.6	Applying implemented fur shader for chipmunk model. Grooming from (Stavginski, 2017).	31
4.7	High quality renders.	32
5.1	A default look of result, produce by Verify BSDF node.	33
5.2	Example of simulation wet fur, (Lin et al., 2011).	34
5.3	From left to right: MIS Bias -1, 0 and 0. Internal mantra’s multiple importance sampling algorithm is used while varying MIS Bias parameter of environment light.	34
A.1	Uploading converted cyHair file to Houdini	41

List of Tables

2.1	Table of Notation for Hair Shading Model, based on (Marschner et al., 2003)	6
2.2	Additional Table of Notation for Fur Shading Model, based on (Yan et al., 2017) and (Yan et al., 2015)	10
2.3	Azimuthal Attenuation from (Yan et al., 2017)	12
2.4	Evaluation and Sample Functions Arguments	14
2.5	Evaluation Function Input Parameters	14
2.6	Evaluation Function Output Parameters	14
2.7	Sample Function Input Parameters	14
2.8	Sample Function Output Parameters	14
3.1	Longitudinal roughness variance for different lobes	20

CHAPTER 1

INTRODUCTION

Fur and hair are ubiquitous in a modern virtual world. It was proven, that visual appearance of hair is the most important personality feature of characters in computer generated content (Ducheneaut et al., 2009). However, despite the fact that the first hair shading model was presented more than two decades ago in (Kajiya and Kay, 1989), the unified efficient algorithm for rendering hair or fur still does not exist. This may be explained by the reason, that it is enormously complex to imitate the interaction of light and hair volume. In addition to the fact that hair rendering is computationally expensive task, it is also difficult to be artistically controlled. This happens due to lack of user friendly parameters.



Figure 1.1: Usage of furry computer generated characters in one of the recent animated feature, (Peter Rabbit 2018)

Many aspects of hair and fur production, including modelling, or styling, and simulation need to be studied to make hair and fur behaviour believable. This research will be focused on the aspects of physically plausible shading for hair fibres. However, it has to be mentioned that other areas have to be comprehensively studied in order to produce natural hair or fur, but they are their own topics of research.



Figure 1.2: Examples of fur, produced by DreamWorks Fur Motion System – Skunk, (Augello and Somasundaram, 2019)

Presented thesis is structured as follows. A brief history of hair rendering is presented in Chapter 2. The details of implementation are described in Chapter 3. This is followed by discussion of results in Chapter 4 and suggestions for further work in Chapter 5.

1.1 Motivation

As it was mentioned, most of the recent production hair shading models suffer from the lack of artistic controls. For their parametrisation are used physical properties of hair instead of user friendly controls, which would directly influence appearance. Because of this, it is extremely difficult to benefit from this hair shading models.

Despite the fact, that there were already exist some attempts, such as Sadeghi et al. (2010), to implement artistically friendly hair shading tool, non of them were implemented in Houdini. Moreover, implemented in Sadeghi et al. (2010) method was *ad hoc* and far from physical plausibility. Nowadays, Monte Carlo method in simulation actual light behaviour dominate over non-physically based approaches, so production rendering is moving away from *ad hoc* methods and most of the old implementations, including Sadeghi et al. (2010), start to be less relevant.

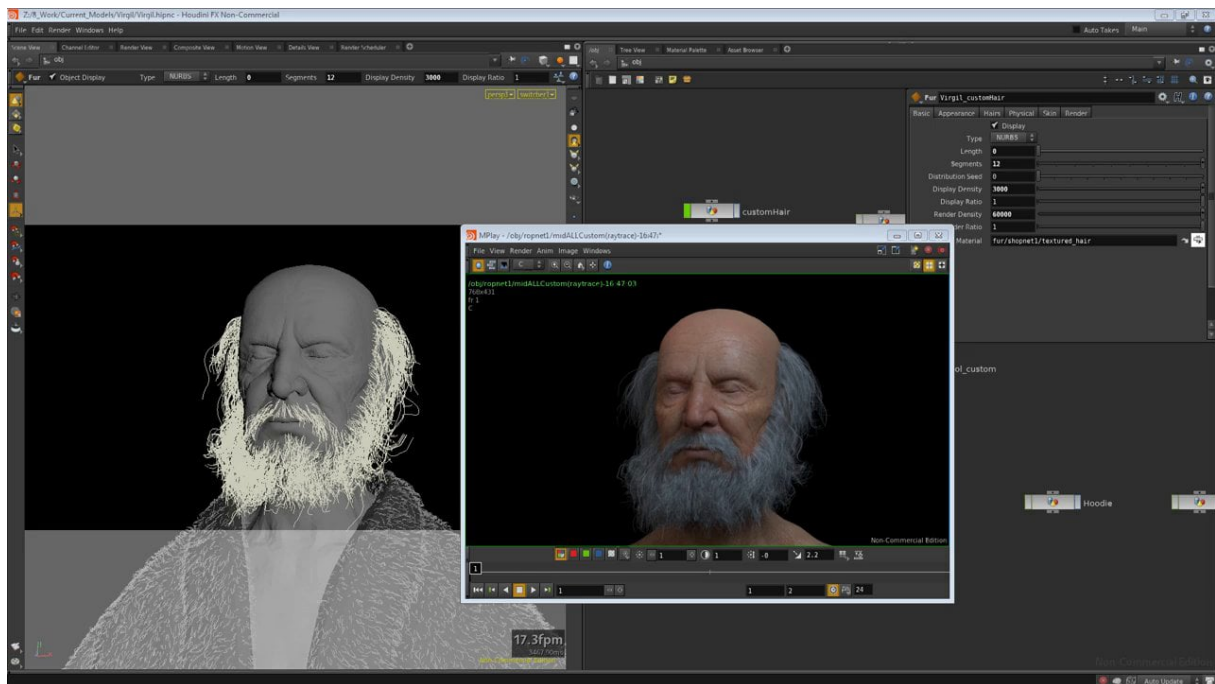


Figure 1.3: Example of grooming in SideFX Houdini, (Browne, 2014)

Currently, there are a wide choice of production renderers: from the most recent GPU-based Redshift to well-known V-Ray. However, due to the boost up the capacities and abilities, production starts to move away from old softwares, which are not able to cope with heavy scenes, to the new, procedurally-based softwares. Houdini is one of the most powerful procedurally-based software, which is capable to

solve approximately all computer graphics tasks. Despite the fact, that it is mainly used for FX, Houdini provides users abilities to do complex modelling, shading, lighting and even compositing. At the time of writing this thesis, the number of followers of Houdini Facebook page was close to fifty thousand (Houdini, 2007), which show increasing popularity of the procedural methodology. Interest in Houdini from look development and lighting point of views has increased after SideFX's announcement on SIGGRAPH 2019 about Solaris, which will bring the new lighting context into Houdini, (SideFX, 2019).

As it was mentioned earlier, Houdini predominantly used for FX, though, Mantra, which is internal Houdini renderer, provides not the less capacities than more popular RenderMan. However, most of its abilities are not well studied.

The main motivation behind the topic, was desire to create custom physically based hair and fur shaders with artistic friendly parametrisation. The intention to use Houdini is explained by the reason, that the usage of Mantra (or Karma in the future) as a main renderer starts intensifying in production.

CHAPTER 2

LITERATURE REVIEW AND THEORETICAL BACKGROUND

This chapter focuses on previous works which were done in an area of physically based hair rendering. It also includes a brief overview of existing approaches for hair shading in different softwares.

2.1 Anatomy

In a real life, human hair have a shape of rough cylinder, which is composed of protein and inactive cells. Internally, each strand includes three main components: cuticle, cortex and medulla, following from outermost to the innermost.

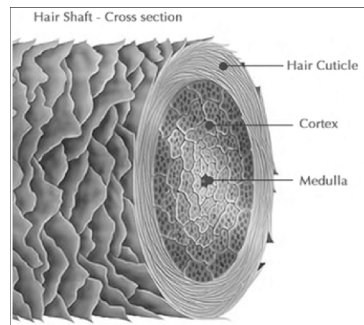


Figure 2.1: Cuticle, cortex and medulla layering, (Morganti and Yuan Li, 2015)

Cuticle consists of overlapping flat cells, which create protecting shield around inner elements of the fibre. Cortex is the thick homogeneous layer of human fibre. Exactly this layer contains significant amount of melanin pigments such as eumelanin or pheomelanin. Eumelanin gives hair blonde, brown or black tones, while pheomelanin gives ginger. Cortex accounts approximately 90% of fibre volume.

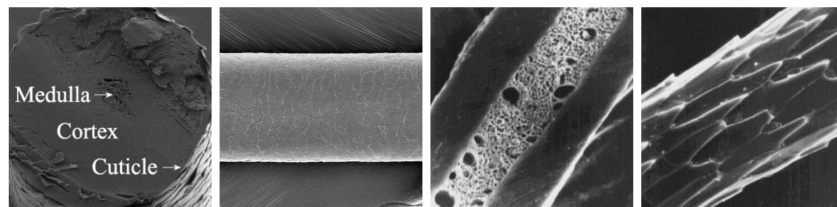


Figure 2.2: From left to right: structure of human hair; cuticle layer on human hair fibre; example of fur fibre; cuticle layer on animal fur fibre, (Galatík et al., 2011), (Wei, 2006)

Medulla is the core of the hair fibre and it may be not as rich of pigment as cortex. Human hair medullary index, which is a ratio of the diameter of the medulla to the fibre, equals around 0.33. Fur fibres at the same time tend to have medullary index between 0.5 and 0.9. So, from this comparison it

can be seen that in contrast to fur, in human hair medulla do not play the crucial role in absorption. Finally, it should be mentioned, the hair and fur are dielectrics.

2.2 Literature review

2.2.1 Physically Based Rendering

First of all, here will be explained some common formalisms of physically based rendering (PBR) approach and theory of radiance. Implementing PBR Model involves creating a bidirectional scattering distribution function (BSDF), where *Bidirectional Scattering Distribution Function* (BSDF) is a probability function which describes general scattering. At the same time, in some literature may be used another term – Bidirectional Reflectance Distribution Function. *Bidirectional Reflectance Distribution Function* (BRDF) is a function which determines «how much energy is reflected from incoming direction ω_i to an outgoing direction ω_o » (Pharr et al., 2016, p.11-12).

The standard reflected radiance equation may be presented as:

$$L_o(\omega_o) = \int_{\Omega} L_i(\omega_i) f_s(\omega_i, \omega_o) \cos(\theta_i) d\omega_i \quad (2.1)$$

Moving on to the hair reflectance theory, in (Zinke and Weber, 2007) was introduced a definition of Bidirectional Curve Scattering Distribution Function. *Bidirectional Curve Scattering Distribution Function* (BCSDF) is a function which perform how light interacts with a single fibre and which encodes the ratio of the irradiance from ω_i to the radiance toward ω_o . For perfect cylindrical geometry, a BCSDF can be referred to general BSDF, integrated over the fibre's width.

2.2.2 Hair Shading Models

In the following section will be discussed the scattering behaviour, which give hair fibres its distinctive appearance.

The large amount of works in an area of physically based rendering are devoted to reflectance hair models. The first of it was presented in (Kajiya and Kay, 1989). In this paper authors developed a reflectance model by considering cylinders that exhibit Lambertian and Phong reflections. In this implementation diffuse component was depended on the longitudinal angle and a reflectance component produced a highlight effect. Azimuthal angle did not take part in this implementation, so this model was azimuthal-indepent. In an equation form, hair shading model, presented in (Kajiya and Kay, 1989), may be written as:

$$f_s(\omega_i, \omega_o) = k_d + k_s \frac{\max[\cos^p(\theta_i + \theta_o), 0]}{\cos(\theta_i)}, \quad (2.2)$$

where k_d is diffuse reflectance, k_s is the specular reflectance, p is the Phong exponent.

Presented by Kajiyaya-Kay hair shading model was improved in (Goldman, 1997) by giving it different opacity values depending on viewing angle.

The main drawback of the shading model, which was presented by Kajiyaya-Kay was the fact, that this model was not energy conserving, nor physically based. However, even despite this facts, due to its simplicity, model, which was presented in (Kajiyaya and Kay, 1989) remained widely used for a decades.

The factored-lobe hair model, which separates reflectance into individual lobes, was firstly presented in (Marschner et al., 2003). General hair shading model from (Marschner et al., 2003) included three lobes, based on number of internal reflections within the fibre cylinder:

- R lobe is responsible for direct reflection of light from the surface of a hair strand, which in other words may be named as *primary highlight*.
- TT lobe is responsible for transmission of light through the hair strand which in other words may be named as *absorption*.
- TRT lobe is the secondary reflection of light traveling through the hair twice before going back.

Table 2.1: Table of Notation for Hair Shading Model, based on (Marschner et al., 2003)

ω_i	Incoming direction
ω_o	Reflected (outgoing) direction
L	Radiance
$f_s(\omega_i, \omega_o)$	BCSDF function
v, w	Normal plain
u	Tangent to the hair in the direction of growth, from root to tip
ϕ_i, ϕ_o	Azimuthal inclinations from the normal plane, $\phi_i, \phi_o \in [-\pi, \pi]$
θ_i, θ_o	Longitudinal inclinations from the normal plane, $\theta_i, \theta_o \in [-\frac{\pi}{2}, \frac{\pi}{2}]$
ϕ	Azimuthal difference between the incident and reflected directions
p	Number of internal path segments, which were traveled by light ray in the hair fibre
$M_p(\theta_i, \theta_o)$	Longitudinal scattering function
$N_p(\theta_i, \theta_o, \phi)$	Azimuthal scattering function
$A(p, h)$	Attenuation function
$S(\theta_i, \theta_o)$	A factored reflectance model
η	The relative index of refraction of the hair
σ_a	Absorption cross-sections for eumelanin in cortex

The total reflectance function S is the sum of reflection functions for each lobe S_p , which responds to p -number internal reflections within the hair fibre:

$$S(\theta_i, \theta_o, \phi) = \frac{dL_o(\omega_o)}{dL_i(\omega_i)} = \sum_{p=0}^{\infty} S_p(\theta_i, \theta_o, \phi) \quad (2.3)$$

$$p \in \{R = 0, TT = 1, TRT = 2, TRRT = 3 \dots\}$$

In general, number of lobes tends to be no more than three, because of to the fact, that higher bounces are largely attenuated and their contribution to total energy is negligible.

Each lobe was presented as a product of two 1D profiles: the longitudinal and the azimuthal scattering functions. In some paper, attenuation function may be separated from azimuthal function. Both, incident and exitant directions are measured in a spherical coordinates.

$$S_p(\theta_i, \theta_o, \phi) = M_p(\theta_i, \theta_o)N_p(\theta_i, \theta_o, \phi) \quad (2.4)$$

Method, which was presented by (Marschner et al., 2003) was extended in various papers. Initially, presented method was not energy conserving, however this was fixed in (d'Eon et al., 2011). Moreover, in (d'Eon et al., 2011) authors presented approach, which allowed to go beyond three lobes: TRRT and etc. To preserve correct evaluation of global illumination, an importance sampling method has to be used for reflectance model. An importance sampling scheme for original (Marschner et al., 2003) model was proposed in (Hery and Ramamoorthi, 2012) and (Ou et al., 2012).

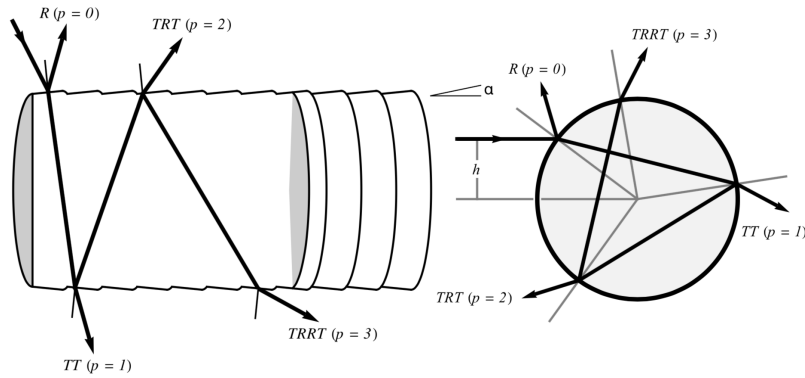


Figure 2.3: A scheme of hair model for one fibre from (Marschner et al., 2003)

In a contrast to longitudinal scattering M_p , azimuthal scattering N_p highly varying across hair fibre's width. So, in (d'Eon et al., 2011) was proposed a solution to azimuthal function, which included integration of the hair fibre's width:

$$N_p(\phi) = \frac{1}{2} \int_1^{-1} A(p, h)D(\phi - \Phi(p, h))dh, \quad (2.5)$$

$$\Phi(p, h) = 2p\gamma_t - 2\gamma_i + p\pi, \quad (2.6)$$

where Φ is the net change in azimuthal direction, $\gamma_i = \arcsin(h)$, $\gamma_t = \arcsin(\frac{h}{\eta'})$ and D is the Gaussian detector, which is also may be known as wrapped Gaussian distribution.

Usage of Gaussian detector imposed significant restrictions on presented hair shading model. In

(d'Eon et al., 2013) was proposed 70-point Gaussian quadrature to capture azimuthal scattering across hair fibre width. However, in case of global illumination, calculation for each shading point this integrand over 70 times is computationally expensive task. Moreover, in (d'Eon et al., 2013) were used precomputed values, which is impractical in real production. So, wrapped Gaussian can be written as:

$$D(\phi) = \sum_{k=-\infty}^{\infty} g(\beta, \phi - 2\pi k) \quad (2.7)$$

where g is a Gaussian distribution, which allows to imitate azimuthal deviation due to surface roughness, and β is standard deviation.

It has to be mentioned, that Gaussian distribution can not be analytically solved, so in most of the calculations is used two-dimensional sampling with usage the Box-Muller transformation.

Initially, all BCSDF represented hair shading model was *far-field*, which means it was assumed that the width of hair fibre is insignificant in a contrast to its length and it is less than a pixel. In (Zinke and Weber, 2007) authors introduced a near-field model for close up rendering of hair.

For computer graphics tasks scientific accuracy is not as important as a final aesthetically pleasing visual appearance. Initial hair shading model, which was presented in (Marschner et al., 2003) was physically accurate, but not artistic friendly. The first simplification of the model from (Marschner et al., 2003) for artistic needs was presented by (Sadeghi et al., 2010). This model was based on the single-scattering model of (Marschner et al., 2003) and the multiple scattering approximation of Dual Scattering (Zinke et al., 2008).

The most recent achievements in implementing both physical correctness and artistic friendliness for hair shading model was presented in (Chiang et al., 2016). In this paper, authors implemented energy conserving model, which was based on (d'Eon et al., 2011) and importance sampling scheme from (d'Eon et al., 2013). However, in a contrast to integral solution for azimuthal scattering, which was presented in (d'Eon et al., 2013), (Chiang et al., 2016) proposed simplified model of azimuthal function, which relies on solution supplied by Monte Carlo rendering system:

$$N_p(\phi, h) = A_p(h)D_p(\phi - \Phi(p, h)) \quad (2.8)$$

Moreover, in (Chiang et al., 2016) was proposed the novel replacement of computationally expensive wrapped Gaussian by more efficient logistic distribution, which is in a contrast to wrapped Gaussian, is analytically integrated over arbitrary finite domain. Reparameterized logistic distribution l_g , which matches the peak of Gaussian distribution, and azimuthal scattering function $D(\phi)$ may be written as:

$$l(x, a) = \frac{e^{-x/s}}{s(1 + e^{-x/s})^2} \quad (2.9)$$

$$l_f(x, s, a, b) = \frac{1}{\frac{1}{1 + e^{a/w}} + \frac{1}{1 + e^{b/w}}} l(x, s) \quad (2.10)$$

$$l_g(x, s, a, b) = l_f(x, s, \sqrt{\frac{\pi}{8}}, a, b) \quad (2.11)$$

$$D(\phi) = l_g(\phi, s, -\pi, \pi) \quad (2.12)$$

It was shown, that in real production usage logistic function instead of 70-point Gaussian quadrature allows to achieve 20x speed up on furry character, which makes brute force path tracing of hair and fur achievable in production rendering.

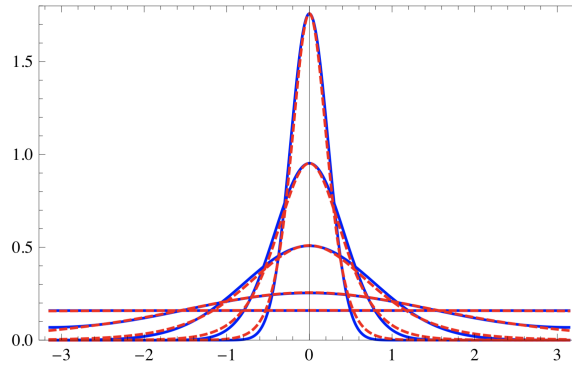


Figure 2.4: Logistic (red) and wrapped Gaussian (blue) distributions, (Chiang et al., 2016)

However, the main contribution from (Chiang et al., 2016) was done in an area of unifying physical and artistic parameters. Instead of using variance v , which was proposed in (d'Eon et al., 2013), in (Chiang et al., 2016) was presented uniform longitudinal roughness $\beta_M \in [0, 1]$, where 0 equals to dielectric and 1 to isotropic. The perceptual experiment were done to define constants in this inverse mapping, which may be presented in a form:

$$v = (0.726\beta_M + 0.812\beta_M^2 + 3.7\beta_M^{20})^2 \quad (2.13)$$

The same simplification was performed for azimuthal roughness β_N and logistic scale factor s . The azimuthal roughness behaves similarly to phase function, which means, that with lower azimuthal roughness hairs are forward scattered and with higher roughness – backward scattered. Calculation of constants in presented equation was done by defining relationship between longitudinal and azimuthal functions through «numerical scattering simulation on cylinders with isotropic microfacet distributions of differing

roughness» (Chiang et al., 2016, p.279).

$$s = 0.265\beta_N + 1.194\beta_N^2 + 5.372\beta_N^{22} \quad (2.14)$$

Finally, non-artistic friendly absorption coefficient σ_a was remapped by multi-scattering albedo C and already presented azimuthal roughness β_N . Constants were achieved after rendering number of hair cubes with various σ_a and measuring value C over the centre of this cubes.

$$\sigma_a = \left(\frac{\ln C}{5.969 - 0.215\beta_N + 2.532\beta_N^2 - 10.73\beta_N^3 + 5.574\beta_N^4 + 0.245\beta_N^5} \right)^2 \quad (2.15)$$

Presented in (Chiang et al., 2016) hair shading model was adopted by Walt Disney Animation Studio and was used on projects such as (Zootopia 2016) and tests were done in Disney's proprietary inhouse Hyperion renderer (Burley et al., 2018).

2.2.3 Fur Shading Model

The innovative approach in rendering fur was presented in (Yan et al., 2015). In this paper authors introduced double cylinder model to represent inner medulla scattering layer. In addition to three simple lobes – R, TT, TRT – authors added *scattered* (by medulla) lobes. Initially, from (Yan et al., 2015), fur model included 8 lobes:

$$p \in \{R, TT, TRT, TrT, TttT, TtrtT, TrRrT, TtRttT\}$$

The validation of presented double-cylindere model was done by measuring the real fur fibres.

Table 2.2: Additional Table of Notation for Fur Shading Model, based on (Yan et al., 2017) and (Yan et al., 2015)

κ	Medullary index
$\sigma_{m,a}$	Absorption coefficient in medulla
$\sigma_{m,s}$	Scattering coefficient in medulla
l	Layers of cuticle scale
s_c	Distance traveled within cortex
s_m	Distance traveled within medulla

The BCSDf for fur shading model from (Yan et al., 2015) may be written as:

$$S(\theta_i, \theta_o, \phi_i, \phi_o, h) = \frac{\sum_p M_p^u(\theta_i, \theta_o) N_p^u(h, \phi)}{\cos^2 \theta_i} + M^s(\theta_i, \theta_o, \phi) \frac{\sum_p N_p^s(h, \phi)}{\cos^2 \theta_i} \quad (2.16)$$

where u and s specify unscattered and scattered (by medulla) lobes respectively.

For unscattered lobes computation of longitudinal and azimuthal scattering functions is performed via Gaussian distribution, which is similar to other implementations:

$$M_p(\theta_i, \theta_o) = g(\theta_o, -\theta_i + \alpha_p, \beta_p) \quad (2.17)$$

$$N_p(\phi, h) = A_p(h)D_p(\phi, h) \quad (2.18)$$

$$D_p = g(\beta_p, \phi - \Phi(p, h)) \quad (2.19)$$

where α_p is the lobe shift and β_p is the roughness. In accordance to (d'Eon et al., 2013), $\alpha_R = 2\alpha$; $\alpha_{TT} = -\alpha$; $\alpha_{TRT} = -4\alpha$, where α is the scale tilt angle.

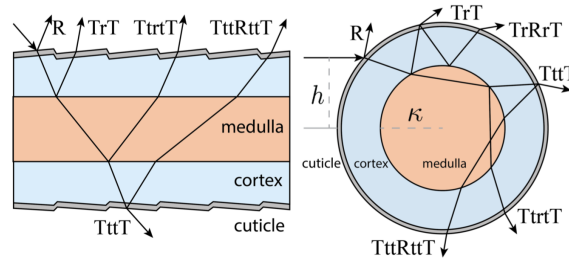


Figure 2.5: All scattered (by medulla) paths in fur shading approach, (Yan et al., 2015)

For scattered lobes (Yan et al., 2015) proposed pre-calculated scattering profiles, which are stored in 4D tables named C^M and C^N .

$$M^s(\theta_i, \theta_o, \phi) = \mu F_t \text{lerp}(C_{\phi=0}^M(\theta'_i, \theta'_o), C_{\phi=\pi}^M(\theta'_i, \theta'_o), \phi) \quad (2.20)$$

$$D_p^s(\phi, h) = C^N(\phi - \Phi_p^s(h)) \quad (2.21)$$

where μ is normalisation factor, F_t is Fresnel transmission through cuticle, θ'_i and θ'_o are incident and outgoing angles that entered and exited medulla respectively.

Presented in (Yan et al., 2015) approach suffered from computational expensiveness and lack of intuitive controls. Also, usage of precomputed data made spatial variation of some parameters ineffable. Number of optimisations to initial model was introduced in (Yan et al., 2017). For example, unified indices of reflection for cortex and medulla allowed to decrease number of lobes from eight to five.

$$p \in \{R, TT, TRT, TT^S, TRT^S\}$$

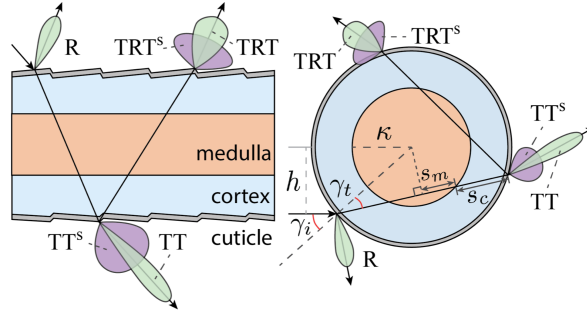


Figure 2.6: Optimised fur lobes, (Yan et al., 2017)

In (Yan et al., 2017) longitudinal scattering function was simplified, due to the reason that in the most practical cases the Fresnel transmittance F_t cancels the normalisation coefficient μ :

$$M^s(\theta_i, \theta_o, \phi) = \text{lerp}(C_{\phi=0}^M(\theta_i, \theta_o), C_{\phi=\pi}^M(\theta_i, \theta_o), \phi) \quad (2.22)$$

Table 2.3: Azimuthal Attenuation from (Yan et al., 2017)

R	F
TT	$(1 - F)^2 \exp\left(-\frac{2s_c\sigma_a + 2s_m(\sigma_{m,a} + \sigma_{m,s})}{\cos\theta_d}\right)$
TRT	$(1 - F)^2 F \exp\left(-\frac{4s_c\sigma_a + 4s_m(\sigma_{m,a} + \sigma_{m,s})}{\cos\theta_d}\right)$
TT^s	$F \exp\left(-\frac{(s_c + 1 - \kappa)\sigma_a + \kappa\sigma_{m,a}}{\cos\theta_d}\right)$
TRT^s	$(1 - F)F \exp\left(-\frac{(3s_c + 1 - \kappa)\sigma_a + (2s_m + \kappa)\sigma_{m,a} + 2s_m\sigma_{m,s}}{\cos\theta_d}\right)$

Despite the fact, that in (Yan et al., 2017) authors proposed tensor compression to precomputed data, which allowed to decrease files size from 600MB to 150KB, usage of pre-calculated data puts a heavy burden on memory, speed and storage, which significantly limiting presented fur reflectance model. Moreover, measurement were done for just some limited animal species, including cat, deer, dog, mouse, raccoon and rabbit.

Presented in both (Yan et al., 2017) and (Yan et al., 2015) renders were implemented within Mitsuba (Jakob, 2010).

2.3 Houdini and VEX

In addition to theoretical background related to the theory of hair rendering, practical aspects of chosen software have to be discussed. VEX is Houdini international language, which performance is close to C/C++ and RenderMan OSL (RSL) shading language. VEX may be used in several places in Houdini, including simulations, modelling, channel operations (CHOPs) and rendering. The internal Houdini renderer Mantra uses VEX for all shading computations, including surface, displacement, light and fog shaders.

However, VEX has some number of limitations, for example, it is not possible to implement recursion within VEX. Instead of this, shader call via importing function has to be used.

Listing 2.1: Example of importing the fur shader

```
import fur;
```

Moreover, similarly to RenderMan shading language OSL, all parameters in VEX are passed by reference. Parameters may be switched to read-only mode by adding *const* prefixing.

The main limitation of VEX lies in an area of arrays. Currently VEX does not support multi-dimensional arrays, which imposes significant restrictions on the manner of writing code. Moreover, arrays can not be passed between shaders.

There are two approaches in building materials in Houdini: with usage VOPs and with usage VEX. While VOPs tend to be considered as more artistic friendly, VEX is more developers and TDs level.



Figure 2.7: Example of VOPs network in shader context in Houdini

VEX includes specific context – CVEX – which unlike VOPs runs at the render time and allows to deal more efficiently with large amount of data, such as hair. To define a custom BSDF reflectance function in Houdini has to be used specific VEX function *cvox_bsdf*, which works via a pair of evaluation and sample functions. Both this functions have to be implemented as a two separate *.vfl* files and have to be stored in *vex/CVex* Houdini configuration folder. Moreover, CVEX is very responsive to naming

conventions, so the name of class and name of the file title for both sample and evaluation functions have to be the same.

Table 2.4: Evaluation and Sample Functions Arguments

Table 2.5: Evaluation Function Input Parameters

- *u* – outgoing light direction
- *v* – incoming light direction
- *bounces* – specification of a lobe, to which evaluated shader will belong, e.g. diffuse, specular, etc
- *reverse* – evaluation bsdf from the camera (0) of from the light (1)

Table 2.6: Evaluation Function Output Parameters

- *refl* – vector value of the reflectivity of the BSDF function, which will be returned by albedo function in a case of its calling
- *eval* – vector value of the reflectance for the given direction
- *pdf* – float value of the sampled pdf for the given direction

Table 2.7: Sample Function Input Parameters

- *u* – outgoing light direction
- *sx* – random value in a range [0, 1], correlated with *sy*
- *sy* – random value in a range [0, 1], correlated with *sx*
- *bounces* – specification of a lobe, to which evaluated shader will belong, e.g. diffuse, specular, etc

Table 2.8: Sample Function Output Parameters

- *refl* – vector value of the reflectivity of the BSDF function, which should match the *refl* from evaluation function.
- *v* – light direction from the surface to the light
- *bouncetype* – specific component type selected by sampling
- *pdf* – sampling pdf

The first two arguments of *cvex_bsdf* function have to be a VEX source strings, while the rest arguments are developer-defined and their number may vary. As an example implemented *extended_hair* shader includes eleven arguments, not including the VEX source strings:

Listing 2.2: cvex_bsdf call

```
bsdf $hair_f = cvex_bsdf("extended_hair_eval", "extended_hair_sample",
                        "label", $label, "bouncemasklabels", $mask,
                        "nN", $nN, "tip", $tip, "h", $shift,
                        "eta", $ior, "colour", $colour,
                        "beta_m", $l_roughness, "beta_n", $a_roughness,
                        "alpha", $angle_shift, "lobe", $lobe);
```

The contrast between evaluation and sample functions are in the fact that the main goal of the sampling function is selecting the random direction for the reflection ray. So, while in evaluation function outgoing light direction is known, in sampling function it has to be calculated.

2.4 Hair rendering in production renderers

Currently, most of the production renderers are based on (Marschner et al., 2003). The most modern algorithms use its extensions such as (Pekelis et al., 2015).

RenderMan 19 was the first commercial renderer which started to support physically based approach in hair shading based on (Marschner et al., 2003).

Initial hair shading model in Houdini Mantra included three main lobes: primary reflection, secondary reflection and transmission. Inside parameters for each of the presented lobes user could adjust colour, size, shift, intensity etc.



Figure 2.8: Standard hair shading models in some production render engines. From left to right: Mantra, Arnold and RenderMan.



Figure 2.9: Left: R, TT and TRT lobes (equals to reflect, diffuse and refract) from standard hair shading model in Mantra. Right: direct (left) and indirect (right) diffuse and specular components of standard hair shading model in RenderMan.

The main approach in hair rendering in 3Delight is close to the common idea of using three lobes: R, TT and TRT. However, in a contrast to many other renders, there are no diffuse term in hair shading model in 3Delight.

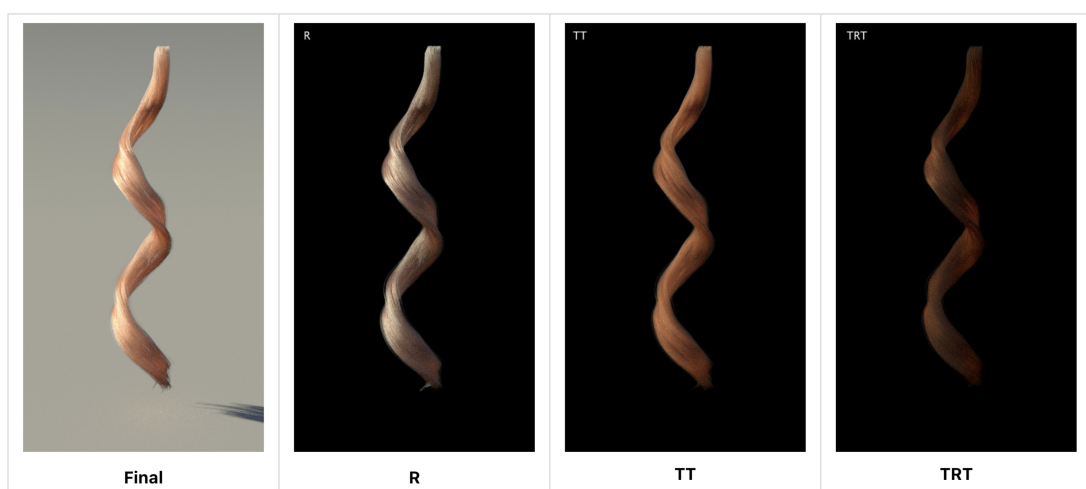


Figure 2.10: 3Delight Hair Model, (DNAResearch, 2013)

Moreover, in addition to commonly used in other renders parameters, 3Delight allows users to adjust some artistic parameters, such as amount of white hairs and switch between natural fur and synthetic fur.

Finally, in a contrast to other renders, 3Delight is proficient both with Yeti and XGen. For example, Arnold Standard Hair shader may correctly works just with Arnold Curve Shapes. Currently, 3Delight integrated in production Look Development and Lighting softwares, including Maya and Katana.

CHAPTER 3

IMPLEMENTATION

The purpose of this project was to investigate the process of implementation BSDF with usage VEX in Houdini and generation of custom hair and fur shaders, based on recent SIGGRAPH papers: (Chiang et al., 2016) and (Yan et al., 2017). This chapter explains the implementation process.

3.1 Pipeline

As it was mentioned before, the process of creation computer generated hair is complex. Creation of photorealistic hair or fur requires knowledges in approximately all aspects of 3D, including modelling, texturing, simulation, etc. So, despite the fact, that the main goal if this project was lighting, shading and rendering CG hair, other steps were also observed within this project.

Basic hair models were taken from Cem Yuksel research (Yuksel, 2006). Initial *cyHair* hair files were presented in a binary format. As a part of the pipeline was created a C++ command line tool, which is called *cyHairConvert*. The aim of this tool was exporting initial *cyHair* files into *.txt* format. This converted file could be loaded in Houdini with usage **cyHairLoad** custom node.

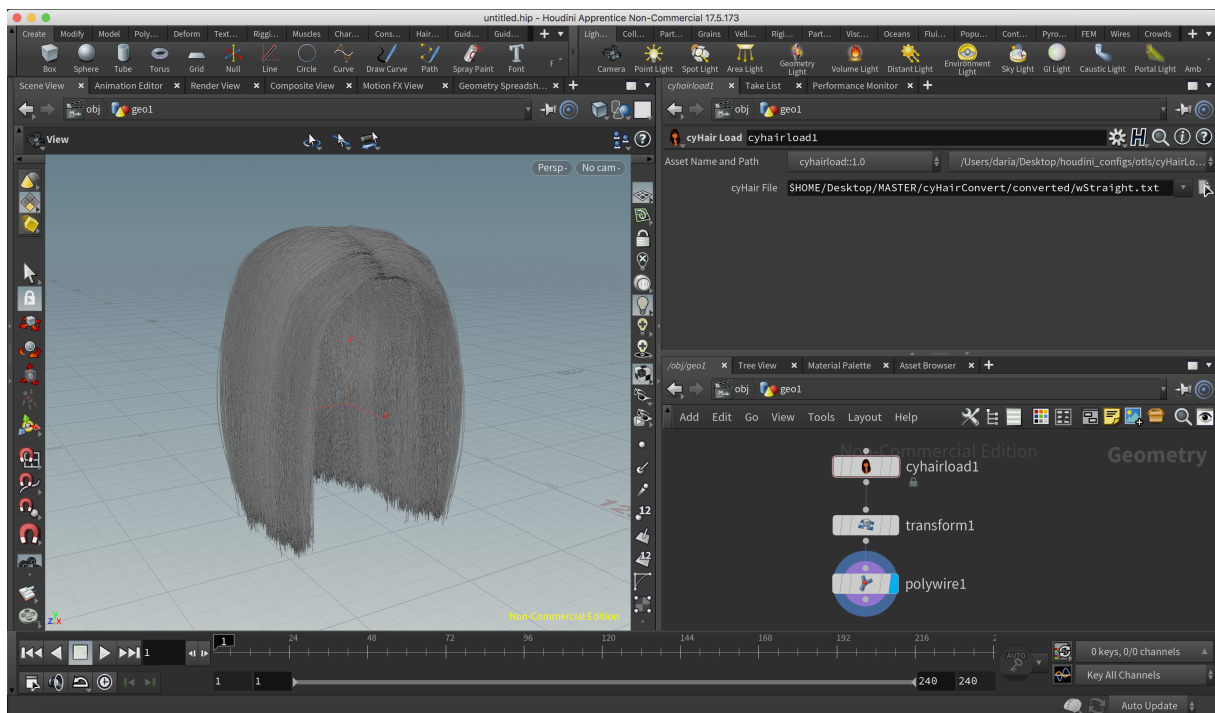


Figure 3.1: CyHairLoad Asset

Python node in described Houdini Digital Asset (HDA) parses converted *.txt* file and creates set of points, connected into strands. There are no default thickness and colour values, however all of them can be defined with usage **Attribute Create** node. For example, thickness value can be controlled with usage standard Houdini attribute *@width*. Colours may be created by *@Cd* attribute and passed to the shader by **Bind** node. Moreover, colour value may be interpolated from root to tip with usage for-each loop and *@ptnum* attribute.



Figure 3.2: Ambient rendering of *cyHair* examples. Models from (Yuksel, 2006)

In a contrast to **cyHairLoad** node, which is a Digital Asset, fur and hair shaders are low level nodes. So they can not be included in Houdini as easily as HDAs. To inject low level node into Houdini it is needed to add new environment variable to Houdini configuration file. Detailed step-by-step explanation of this process can be found in Appendix User Manual.

```

1 writenodes = nuke.selectedNodes('Write')
2 for writenode in writenodes:
3     writenode['Render'].execute()

```

Figure 3.3: Simple Nuke Script to run through nodes in Node Graph

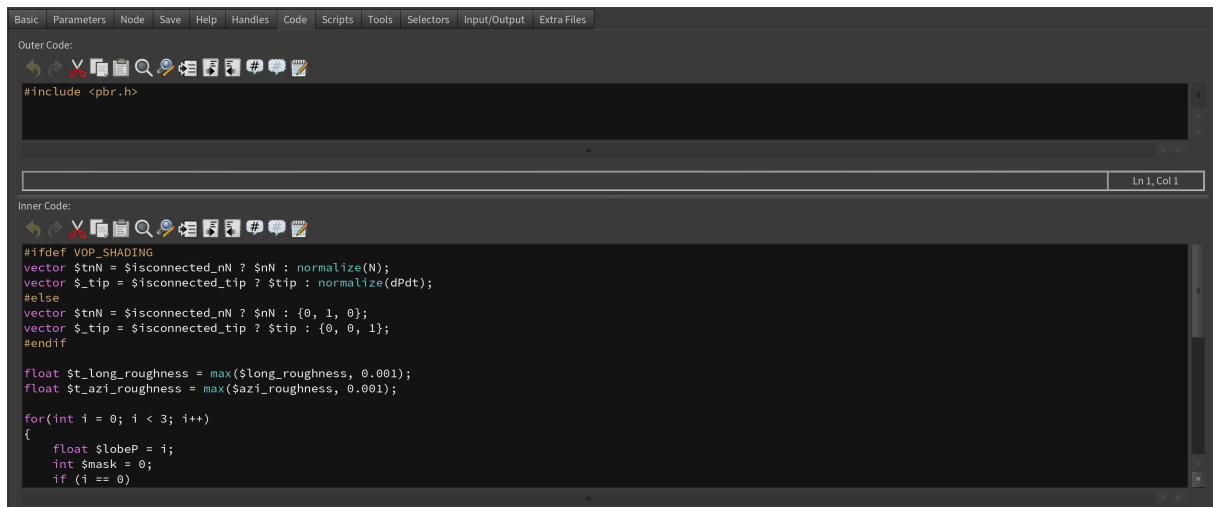
Finally, within process of assembling of all renders was used a simple Python script in Nuke, which ran through all nodes and executed render operation. It was just simple four-lines code, which lightened the workload.

3.2 Shader Implementation

One of the goal of this project was implementation of two shaders: hair shader, based on (Chiang et al., 2016), and fur shader, based on (Yan et al., 2017) and (Yan et al., 2015). It has to be mentioned, that the solution from (Yan et al., 2017) was not been injected into production render before.

Due to the fact, that by default Houdini already supports hair shader, which is based on model from (Marschner et al., 2003), it was decided to add prefix *extended-* to implemented hair reflectance model.

The particular structure of *cvox_bsf* function and both evaluation and sample function were explained in Section 2.3.



```

Outer Code:
#include <pbr.h>

Inner Code:
#ifdef VOP_SHADING
vector $tnN = $isconnected_nN ? $nN : normalize(N);
vector $tTip = $isconnected_tip ? $tip : normalize(dPdt);
#else
vector $tnN = $isconnected_nN ? $nN : {0, 1, 0};
vector $tTip = $isconnected_tip ? $tip : {0, 0, 1};
#endif

float $tLongRoughness = max($long_roughness, 0.001);
float $tAziRoughness = max($azi_roughness, 0.001);

for(int i = 0; i < 3; i++)
{
    float $lobeP = i;
    int $mask = 0;
    if (i == 0)

```

Figure 3.4: Internal code in **PBR Hair Shader Extended** node.

While in default Houdini hair shader each lobe is presented as separate node, inside which is used a wrapper hair CVEX function, in presented solution all lobes are called from asset's internal code. Some basics of this code were taken from original **PBR Hair Primary Reflection** node. An actual *cvox_bsf* function for shader call can be found in Listing 2.2.

3.2.1 Extended Hair Shader

As it was discussed earlier, most of the previous hair reflectance models used physical properties of hair for parametrisation, which is not always artistically friendly. For example, default hair shading node in Houdini has more than fifteen user controls. So, obtain a desire look with such a number of parameters may be challenging. Based on Equations (2.13) - (2.15), in implemented hair shading model was performed reparametrisation of the absorption coefficient and roughness value.

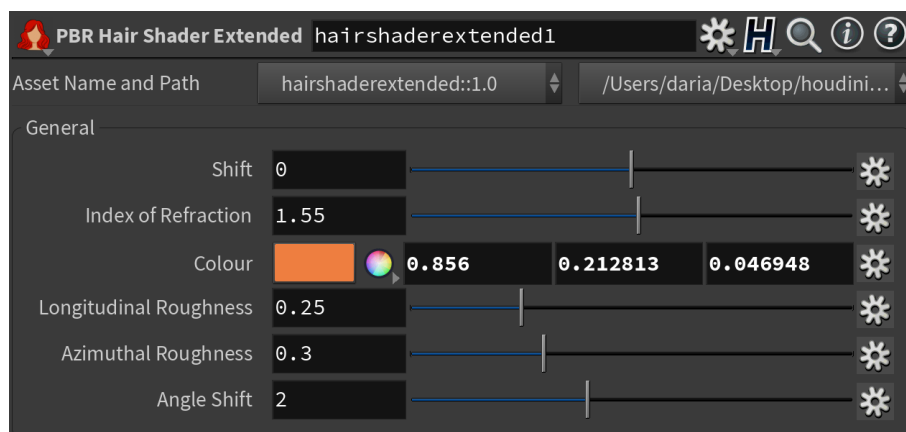


Figure 3.5: **PBR Hair Shader Extended** parameters

So, presented extended hair shader includes just six user parameters: *Shift*, *Index of Refraction*, *Colour*, *Longitudinal Roughness*, *Azimuthal Roughness* and *Angle Shift*.

Finally, it has to be mentioned that for both, **PBR Hair Shader Extended** and **PBR Fur Shader** was written a short help card, which gives a brief explanation of all parameters and some background information, based on this paper and Chapter 2 in particular.

Longitudinal Function

For calculation longitudinal function was used modified Bessel function I_0 of the first kind, based on numerical approximations, which was presented in (d'Eon et al., 2011).

$$M_p(\theta_o, \theta_i) = \frac{1}{2v \sinh(1/v)} \exp\left(-\frac{\sin \theta_i \sin \theta_o}{v}\right) I_0\left(\frac{\cos \theta_o \cos \theta_i}{v}\right) \quad (3.1)$$

Moreover, in a contrast to (Chiang et al., 2016), where the longitudinal roughness variance v considered to be the same for all lobes, in (Pharr, 2016) was shown that for second lobe TT this parameter has to be reduces by an empirical factor, while for third lobe TRT it has to be increased.

Table 3.1: Longitudinal roughness variance for different lobes

$p == 0$	v , from Equation (2.13)
$p == 1$	$0.25v$
$p == 1$	$4v$

Attenuation Function

The attenuation function A_p combines two factors: absorption of light that passes through hair, which gives hair a colour, and Fresnel reflection. To calculate this modified index of refraction η' is used.

$$\eta' = \frac{\sqrt{\eta^2 - \sin^2 \theta_o}}{\cos \theta_o} \quad (3.2)$$

To evaluate transmission is used Beer's law:

$$T = \exp\left(-\sigma_a \frac{2 \cos \gamma_t}{\cos \theta_t}\right) \quad (3.3)$$

Azimuthal Function

Following (Chiang et al., 2016), for calculation azimuthal scattering was used logistic distribution, based on Equations (2.9) – (2.12).

3.2.2 Fur Shader

The fur shader is implemented in a similar way as extended hair shader. The main difference is an increased number of lobes. In accordance to (Yan et al., 2017) fur in addition to cortex layer, has inner cylinder which is called medulla. Moreover, in addition to six parameters from extended hair shader, to the fur shader were added medulla parameters such as: *Medulla Colour*, *Medulla Scattering*, *Medulla Index* and *Cuticle Layers*. As it can be seen, in a contrast to extended hair shader, fur shader has parameters which are based more on physical properties, rather than on artistic controllability.

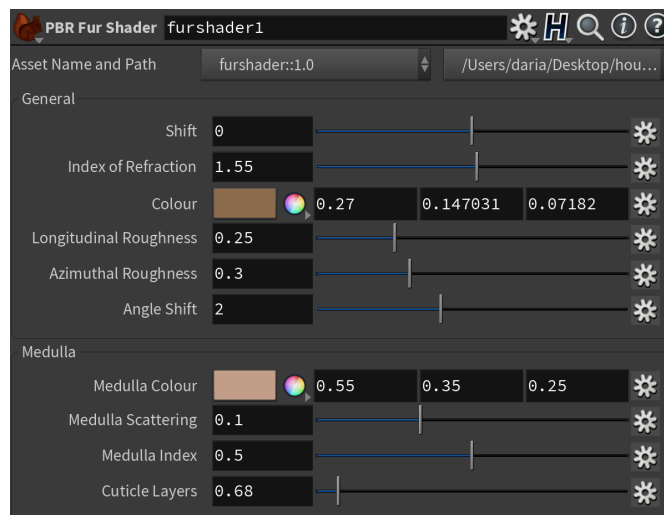


Figure 3.6: PBR Fur Shader parameters

Pre-computed data

As it was described in Section 2.2.3, presented in (Yan et al., 2017) and (Yan et al., 2015) model is based on measurements which are stored in binary files. This method can be easily used in some open source renderers, such as Mitsuba (Jakob, 2010), where it is possible to extend default renderer’s capability by writing additional C++ code. However, production renderers may not provide such options. For example, VEX is significantly limited in some operations, including reading-writing files. So, the method for implemented fur shader is just partly based on (Yan et al., 2017). The main source code was used from extended hair shader, however, there were added some modifications in accordance to two medulla scattered lobes: TT^s and TRT^s .

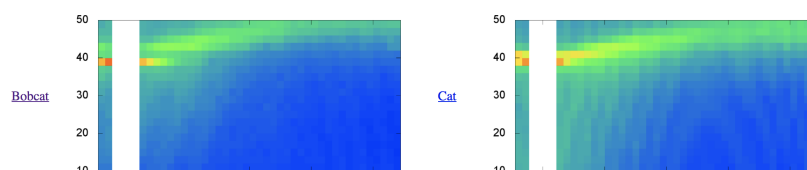


Figure 3.7: The plots of logarithm values of measured raw reflectance data for bobcat and cat, from (Yan et al., 2016).

While working on fur shader, there were done number of tests in injecting this pre-computed files, which are available on (Yan et al., 2016). It was performed with usage C++ code, which converted initial binary files to header *.h* file, the weight of which amounted around 1GB. This file included global function and array of data in it. The total number of elements in final one-dimensional array was around 50 millions. The purpose of the function was returning value from array by index. Produced header file was included into files with sample and evaluation function. This ad hoc method allowed to implicitly read files in VEX. However, even simple including this file was time consuming – it was measured that for start without including header needed around 6 seconds, while for start with included header – 118 seconds. After adding a line with a request of one value from presented array to evaluation function, the render start time increased to 15 minutes, so this approach was considered to be inappropriate.

3.3 Render Adjustment

Shift

The shift allows to control the offset where the ray intersect the oriented fibre. This value in particular is used in Equation (3.3), where $\cos \gamma_t = \sqrt{1 - (\frac{h}{\eta'})^2}$.



Figure 3.8: From left to right: shift -1, 0 and 1.

Longitudinal Scattering

Longitudinal function M_p is responsible for specular along the length of hair, so in other words the longitudinal roughness controls size of the highlight. The lower value, the shine looks more metallic. With high value of longitudinal roughness hair tend to look too flat. The average value for human hair is around 0.25.



Figure 3.9: From left to right: longitudinal roughness 0.1, 0.25 and 0.6. The hair with longitudinal roughness 0.1 looks shiny, while hair with longitudinal roughness 0.6 looks flat and diffuse.

Azimuthal Scattering

The azimuthal function influence the amount of light which is able to exit the hair volume after multiple scattering inside it. Resulting, the hair looks lighter, as the longitudinal roughness increases. This happens due to the fact, that more multiply-scattered light can exit the hair volume.



Figure 3.10: From left to right: azimuthal roughness 0.1, 0.25, 0.5 and 0.7. The hair with azimuthal roughness 0.1 looks darker than hair with azimuthal roughness 0.7.

Angle Shift

The alpha angle α is responsible for offsetting small scales from the cylinder base. When alpha equals zero fibres turn to round ribbons.



Figure 3.11: From left to right: alpha 2 and 4.5.

Absorption and Colour

The calculation of absorption coefficient is performed via Equation (2.15) from (Chiang et al., 2016).



Figure 3.12: Example of various hair colours.

Medulla Index

In fur shader in addition to main six parameters were presented medulla parameters, the main of its is medulla index value. This parameter is responsible for amount of medulla in rendered hair volume and works as a switcher between base colour and medulla colour.



Figure 3.13: Medulla index 0 and 1 correspondingly. When medulla index equals 0, the main colour of the hair is taken from base colour parameter, and when medulla index equals 1 – from medulla colour respectively. In-between values are calculated in accordance to absorption function from Table 2.3.

Medulla Scattering

While in extended hair shader scattering argument was used implicitly, due to its indirect reparametrization in Equations (2.13) – (2.15), in fur shader this parameter is used explicitly.

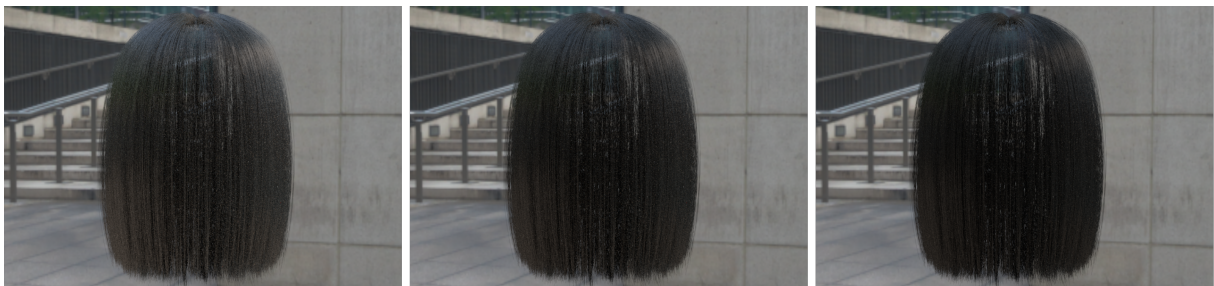


Figure 3.14: From left to right: medulla scattering 0.01, 0.065 and 0.2.

Cuticle Layers

Another additional physical parameter proposed initially in (Yan et al., 2015) is the number of cuticle layers. In accordance to real anatomy of fibre, cuticle forms a layer over the cortex. So, cuticle consists of number of cells stacked up on each other. Cuticle layer property is responsible for increasing cuticle reflectance.



Figure 3.15: From left to right: cuticle layers 0.1 and 2.45.

3.4 Simulation

One of the purpose doing hair simulation was testing motion blur. To achieve motion blur effect after simulation velocity attribute @v has to be preserved and geometry velocity blur has to be activated.



Figure 3.16: Left: renders with motion blur and without motion blur. Right: node network for producing hair simulation.

As it was mentioned before, while working on this projects in addition to main goal, which was writing custom BSDF for hair rendering, in this project was performed a simple hair simulation. It was done with usage vellum solver, which is one of the most recent technique in simulation area. For simulation

was used one of the *cyHair* model, which was converted by the tool, which was described in Section 3.1. The trickiest part was attaching root points to animated head model. This was an issue, due to the fact, that strands were not originally produced in Houdini. Binding points to model was solved with usage **Point Deform** node. The rest of the strand was copied to attached point in accordance to its class attribute, which was achieved from **Connectivity** node, and with usage for-each loop.

3.5 Render Passes and Lobes

The three main lobes may be observed separately by writing them to separate render passes. In terms of fur shader, TT and TT^s as well as TRT and TRT^s are combined to one pass.



Figure 3.17: From left to right: R , TT and TT^s , TRT and TRT^s . Model from (Stavginski, 2017).

3.6 Rendering and Assembling

While working on the project was performed great number of different tests with various shader parameters adjustments. The total number of rendered frames was around six thousands. To assemble all this renders was prepared one Nuke script and three Houdini scenes (with grey shaded hair models, with various shader tests and with chipmunk). All renders were done without background, so background and foreground were merged on compositing stage.

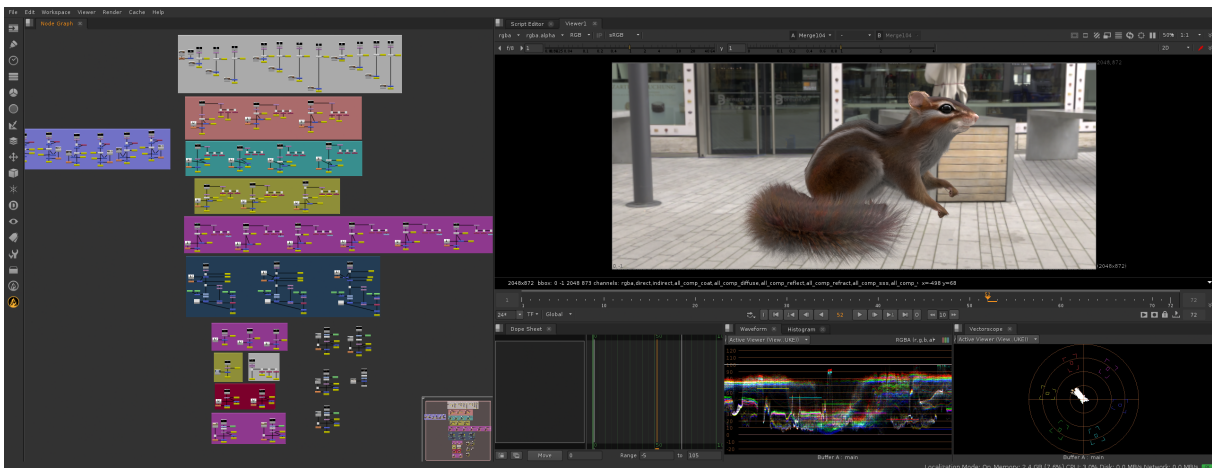


Figure 3.18: Work in progress Nuke script with assembling all renders.

Inside all Houdini render nodes were used expressions, such as refer to node's name. This allowed to save significant amount of time on render setups. Moreover, for all nodes were configured render passes, which allowed to check all lobes, such as R , TT or TRT separately.

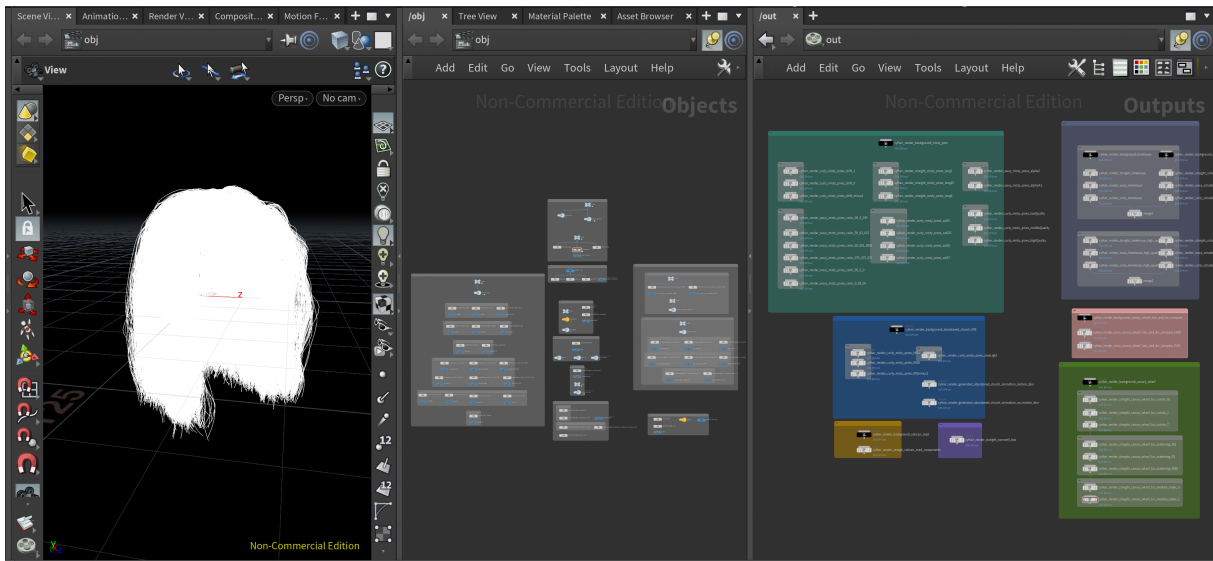


Figure 3.19: Work in progress Houdini scene with various shader tests.

The final video was assembled from videos, which were rendered from Nuke in Apple ProRes 4444 format, in Final Cut Pro.

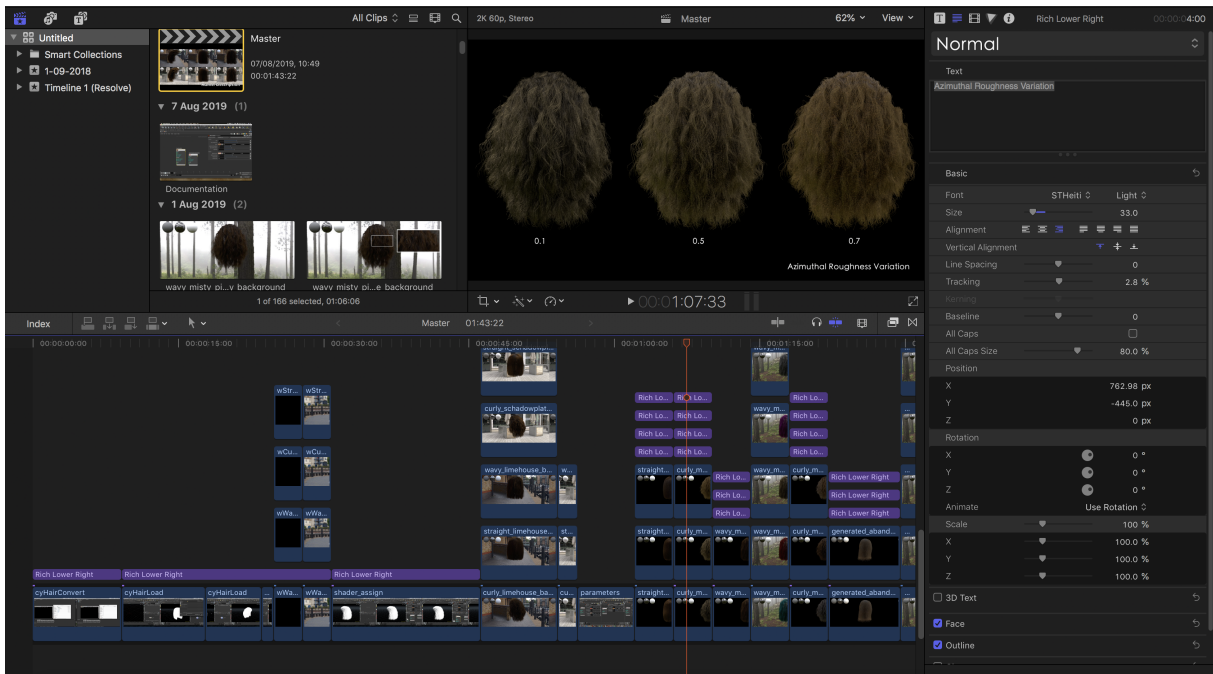


Figure 3.20: Work in progress Final Cut editing.

The final typesetting of presented paper was done in L^AT_EX. The citation and list of references were done with usage B_IT_EX.

Name	^	Date Modified	Size	Kind
▶ folder		11 Jul 2019 at 17:04	--	Folder
▶ appendices		11 Jul 2019 at 17:04	--	Folder
▣ bibliography.bib		Yesterday at 17:11	14 KB	BibTeX
▣ boxit.sty		3 Jul 2019 at 18:35	3 KB	sty
▶ chapters		Today at 11:38	--	Folder
🗑️ comp.nk		Yesterday at 17:08	79 KB	Nuke Project
▣ comp.nk~		Yesterday at 15:59	79 KB	Document
▶ figures		Today at 11:40	--	Folder
▣ mathchars.sty		3 Jul 2019 at 18:35	3 KB	sty
▶ preamble		26 Jul 2019 at 14:57	--	Folder
▣ thesis-mpgraph.mp		5 Jul 2019 at 15:35	5 bytes	mp
▣ thesis.bbl		Yesterday at 16:52	9 KB	Sublim...cument
▣ thesis.bcf		11 Jul 2019 at 17:34	66 KB	Document
▣ thesis.pdf		Today at 11:41	88 MB	PDF document
▣ thesis.preamble.aux		Today at 11:40	985 bytes	aux
▣ thesis.preamble.tex		3 Jul 2019 at 18:35	4 KB	LaTeX
▣ thesis.run.xml		11 Jul 2019 at 17:33	2 KB	XML Document
▣ thesis.sty		12 Jul 2019 at 20:50	3 KB	sty
▣ thesis.tex		3 Aug 2019 at 14:52	3 KB	LaTeX
▣ thesis.top		5 Jul 2019 at 15:35	343 bytes	Document
▣ thesis.tui		5 Jul 2019 at 15:35	Zero bytes	Document
▣ uhead.sty		3 Jul 2019 at 18:35	1 KB	sty

Figure 3.21: Directory structure for L^AT_EX compiling.

So finally, while working on this project were used: Qt Creator, Visual Studio Code, Sublime Text 3, GNU nano, SideFX Houdini, Foundry Nuke, Final Cut Pro and TexShop.

CHAPTER 4

RESULTS

As a result of this project were implemented two custom PBR shaders: for hair and for fur. For practical evaluation capabilities of the shaders were done a number of tests with different shader setups. Hair models were tested under different environment lights, including (Zaal, 2017b), (Zaal, 2017a), (Mischok, 2019b), (Zaal, 2016) and (Mischok, 2019a).



Figure 4.1: Used HDRIs. From left to right: (Zaal, 2017b), (Mischok, 2019b), (Zaal, 2017a), (Mischok, 2019a) and (Zaal, 2016).

4.1 Extended Hair Shader

For **PBR Hair Shader Extended** were performed six test renders: two renders under two different environment lights for three hair model (wavy, curly and straight).



Figure 4.2: Three hair models with same hair shader under two different environment lights.

Render Quality

The render quality of produced images was directly related to number of pixel samples in render node. The number of bounces for each lobe also affected render time, however, it was not as dramatic as total

pixel samples. Moreover, it was observed, that differences in render time and appearance were noticeable just between 0 and 1 bounces for each lobe, while their increasing to 2 or more did not give significant changes. The parameters, such as azimuthal roughness also had affected the amount of flickers. Most of produced renders in this project were done in middle quality due to time limitations.



Figure 4.3: From left to right: low quality, middle quality and high quality.

Global Illumination

For testing global illumination was used a ubiquitous Cornell box scene and area lighting. By this example, it can be seen, that presented hair shader gathering the radiance from coloured walls.

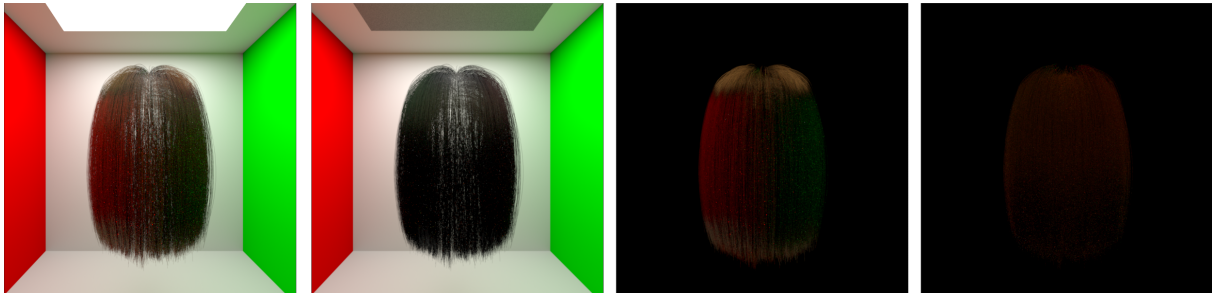


Figure 4.4: The Cornell Box test for extended hair shader and example of separate render passes.

4.2 Fur Shader

PBR Fur Shader is an extension of described previously **PBR Hair Shader Extended**. In case when main colour and medulla colour are close the difference is minor and can be visible just in tones. This may be explained by the fact, that in presented solution was not used pre-computed data from (Yan et al., 2017) and (Yan et al., 2015), which was discussed earlier.



Figure 4.5: Comparison of default hair (left) and fur (right) shaders.

In addition to hair models, to test fur shader was used model and grooming of chipmunk from (Stavginski, 2017). This example also shows the usage texture map as an input for colour parameter.



Figure 4.6: Applying implemented fur shader for chipmunk model. Grooming from (Stavginski, 2017).



Figure 4.7: High quality renders.

CHAPTER 5

KNOWN ISSUES AND FURTHER WORK

5.1 Further Work

During the initial stage of this project the intention was to also implement similar shaders in RenderMan for Houdini, however only Mantra was attempted. Porting the shaders to RenderMan would provide an ability to switch between renders without necessity to adjust shaders from scratch. It will be especially useful in context of capabilities new Karma Renderer. Moreover, implementation of the same algorithm within two different renderers will allow to make a comparative analysis of render speed on the same data with similar calculation in the shader. Currently, the render time, which was shown by Mantra on high settings may be improved, which is also may be named as an area for further research.

From artistic point of view, in addition to parameters which was presented in (Chiang et al., 2016) and (Yan et al., 2017), will be useful to add parameters such as amount of white hair. Also, it may be helpful to give user a choice: use a colour as a main parameter or switch to combination of eumelanin and pheomelanin.

Moreover, while working on this project, it was noticed that Mantra is limited in testing custom BSDFs. Currently, there is a way to check it by node **Verify BSDF**, which was found not obvious. Implementation of user friendly tool for checking albedo, pdf and sampling function will be useful topic for further research.

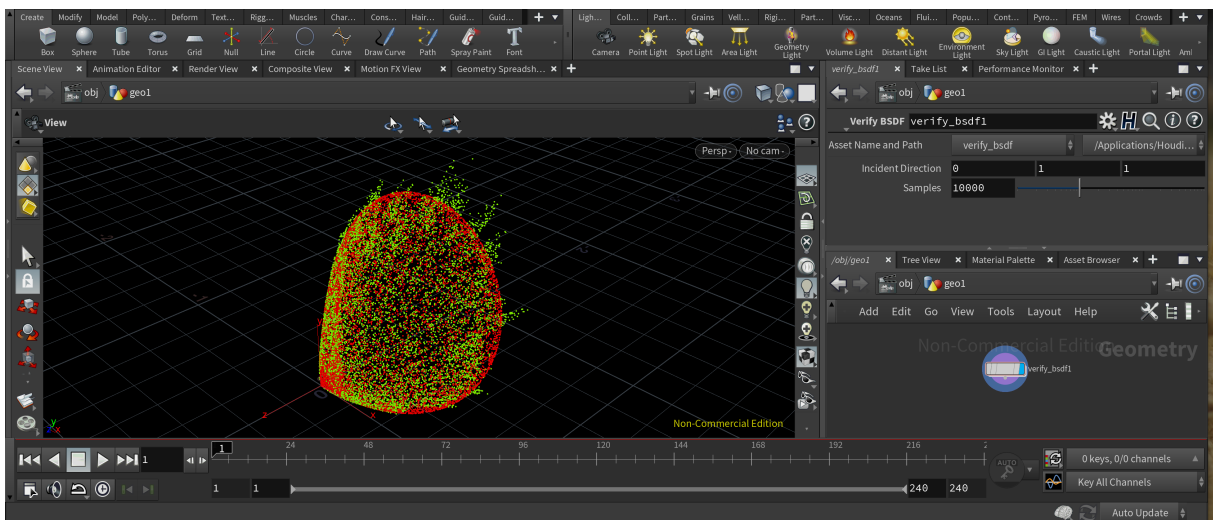


Figure 5.1: A default look of result, produce by **Verify BSDF** node.

Most of the researches are concentrated on dry hair, while the investigation of differences in light

scattering for wet and dry hair is also the broad topic for further investigation. It was already noted that in wet condition all objects tend to look darker and shinier (Jensen et al., 1999) and the hair is not an exception. Recently, there have been done researches in an area of liquid and hair interaction (Lin et al., 2013), (Fei et al., 2017), but non of them concentrated on rendering or shading.

For the future examination of all shaders, the «white furnace» test, which checks that in case when hair does not absorb any light, all the indirect lighting is reflected, (Heitz, 2014) and (Hery and Ramamoorthi, 2012), has to be implemented.

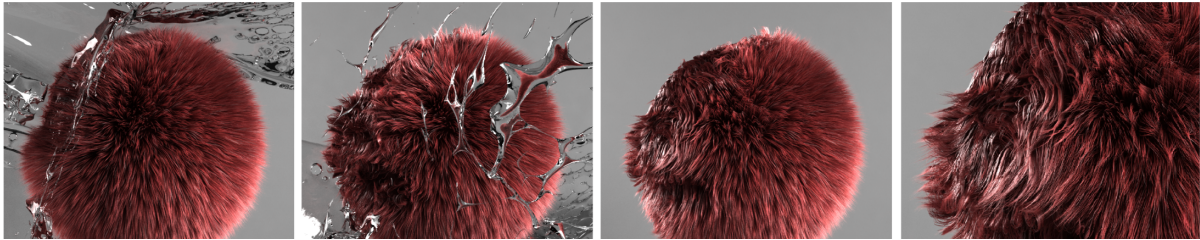


Figure 5.2: Example of simulation wet fur, (Lin et al., 2011).

As it is originally said on Houdini Documentation website, *cvex_bsdf* approach is not ideal at this moment and it is currently in the process of development. So with a new versions of Houdini *cvex_bsdf* can be significantly changed, including the syntax. So, the suggestion for all further researches are better to be done after releasing new lighting LOP operators in Houdini and Karma Renderer, which may replace Mantra in the future.

5.2 Known Issues

As it was said in previous section, currently in Houdini does not exist intuitive way to check written BSDF. While testing different MIS Biases, it was found out that results has some minor differences in appearance. The problem of checking that written BSDF is MIS-weighted may be named as a known issue and a topic for further research at the same time.



Figure 5.3: From left to right: MIS Bias -1, 0 and 0. Internal mantra's multiple importance sampling algorithm is used while varying MIS Bias parameter of environment light.

However, it was checked, that enabling Ray Tracing Background parameter for environment light, does not change the result, which shows the correctness of calculation of *refl* value in sampling function.

CHAPTER 6

CONCLUSIONS

A key objective of this project was implementation of user friendly hair shader with a minimal number parameters. This project was also aimed to show rendering and shading capabilities from TD point of view for not as well investigated in this aspects software.

Moreover, a broad understanding of physically based rendering as well as good mathematical understanding, especially in an area of probability theory, were acquired while working on this project.

In a contrast to already existing shaders, presented in this paper **PBR Hair Shader Extended** can provide a good result approximately out-of-the-box. Implemented shader does not require time for precise tweaking of parameters and understanding an effect from each of it. Moreover, the small number of parameters does not left a margin for error. To sum up, it can be said, that presented in this paper solution follows one of the main principle of the physically plausible paradigm, which is simplicity to use.

To conclude, in this thesis was introduced an implementation of artistic friendly hair shading tool. Both shader are easy to adjust in contrast to already existing ones.

Finally, despite the fact that presented solutions have workable results there are still room for further researches.

Bibliography

- Peter Rabbit*, 2018. [film, DVD]. Directed by Will Gluck. Sony Pictures Releasing.
- Zootopia*, 2016. [film, DVD]. Directed by Byron Howard and Rich Moore. Walt Disney Studios Motion Pictures.
- AUGELLO, N. AND SOMASUNDARAM, A. 2019. Skunk: Dreamworks fur motion system. *In Proceedings of the 2019 Digital Production Symposium, DigiPro '19*, pp. 8:1–8:3, New York, NY, USA. ACM.
- BROWNE, A. 2014. An artist's approach to fur in houdini 13. *Vimeo* [online] <https://vimeo.com/88711732>. Accessed: August 10, 2019.
- BURLEY, B., ADLER, D., CHIANG, M. J.-Y., DRISKILL, H., HABEL, R., KELLY, P., KUTZ, P., LI, Y. K., AND TEECE, D. 2018. The Design and Evolution of Disney's Hyperion renderer. *ACM Trans. Graph.* 37:33:1–33:22.
- CHIANG, M. J.-Y., BITTERLI, B., TAPPAN, C., AND BURLEY, B. 2016. A practical and controllable hair and fur model for production path tracing. *In Proceedings of the 37th Annual Conference of the European Association for Computer Graphics, EG '16*, pp. 275–283, Goslar Germany, Germany. Eurographics Association.
- D'EON, E., FRANCOIS, G., HILL, M., LETTERI, J., AND AUBRY, J.-M. 2011. An energy-conserving hair reflectance model. *In Proceedings of the Twenty-second Eurographics Conference on Rendering, EGSR '11*, pp. 1181–1187, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.
- D'EON, E., MARSCHNER, S., AND HANIKA, J. 2013. Importance sampling for physically-based hair fiber models. *In SIGGRAPH Asia 2013 Technical Briefs, SA '13*, pp. 25:1–25:4, New York, NY, USA. ACM.
- DNARESEARCH 2013. 3delight hair documentation. <https://3delight.atlassian.net/wiki/spaces/3DFM/pages/8388629/3Delight+Hair>. Accessed: July 28, 2019.
- DUCHENEAUT, N., WEN, M.-H., YEE, N., AND WADLEY, G. 2009. Body and Mind: A study of avatar personalization in three virtual worlds. pp. 1151–1160.
- FEI, Y. R., MAIA, H. T., BATTY, C., ZHENG, C., AND GRINSPUN, E. 2017. A multi-scale model for simulating liquid-hair interactions. *ACM Trans. Graph.* 36:56:1–56:17.
- GALATÍK, A., GALATÍK, J., KRUL, Z., AND GALATÍK, A. J. 2011. Furskin identification. <https://www.furskin.cz>. Accessed: August 05, 2019.

- GOLDMAN, D. B. 1997. Fake fur rendering. *In* Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, pp. 127–134, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- HEITZ, E. 2014. Understanding the masking-shadowing function in microfacet-based brdfs. *Journal of Computer Graphics Techniques (JCGT)* 3:48–107.
- HERY, C. AND RAMAMOORTHI, R. 2012. Importance sampling of reflection from hair fibers. *Journal of Computer Graphics Techniques (JCGT)* 1:1–17.
- HOUDINI 2007. Houdini3d. *Facebook* [online]. <https://www.facebook.com/Houdini3D/>. Accessed: July 28, 2019.
- JAKOB, W. 2010. Mitsuba renderer. <http://www.mitsuba-renderer.org>. Accessed: April 12, 2019.
- JENSEN, H. W., LEGAKIS, J., AND DORSEY, J. 1999. Rendering of wet materials. *In* Rendering Techniques.
- KAJIYA, J. T. AND KAY, T. L. 1989. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.* 23:271–280.
- LIN, W. C., LIAO, W.-K., AND LEE, C.-H. 2011. Simulating and rendering wet hair. *In* SIGGRAPH Asia 2011 Posters, SA '11, pp. 39:1–39:1, New York, NY, USA. ACM.
- LIN, W.-C., LIAO, W.-K., AND LEE, C.-H. 2013. Animating wet hair interacting with particle-based fluid. *In* SIGGRAPH Asia 2013 Posters, SA '13, pp. 34:1–34:1, New York, NY, USA. ACM.
- MARSCHNER, S. R., JENSEN, H. W., CAMMARANO, M., WORLEY, S., AND HANRAHAN, P. 2003. Light scattering from human hair fibers. *ACM Trans. Graph.* 22:780–791.
- MISCHOK, A. 2019a. Canary wharf (HDRI Haven). https://hdrihaven.com/hdri/?h=canary_wharf. Accessed: July 28, 2019.
- MISCHOK, A. 2019b. Limehouse (HDRI Haven). <https://hdrihaven.com/hdri/?h=limehouse>. Accessed: July 28, 2019.
- MORGANTI, P. AND YUAN LI, H. 2015. Innovation in cosmetic and medical science. the role of chitin nanofibrils composites. *Journal of Applied Cosmetology* 33:9–24.
- OU, J., XIE, F., KRISHNAMACHARI, P., AND PELLACINI, F. 2012. Ishair: Importance sampling for hair scattering. *In* ACM SIGGRAPH 2012 Talks, SIGGRAPH '12, pp. 28:1–28:1, New York, NY, USA. ACM.

- PEKELIS, L., HERY, C., VILLEMEN, R., AND LING, J. 2015. A data-driven light scattering model for hair.
- PHARR, M. 2016. The implementation of a hair scattering model. <https://www.pbrt.org/hair.pdf>. Accessed: August 7, 2019.
- PHARR, M., JAKOB, W., AND HUMPHREYS, G. 2016. Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.
- SADEGHI, I., PRITCHETT, H., JENSEN, H. W., AND TAMSTORF, R. 2010. An artist friendly hair shading system. *ACM Trans. Graph.* 29:56:1–56:10.
- SIDEFX 2019. SideFX Solaris. <https://www.sidefx.com/community/sidefx-solaris/>. Accessed: August 1, 2019.
- STAVGINSKI, K. 2017. Fur and hair grooming toolset h16.5 masterclass. <https://www.sidefx.com/tutorials/houdini-165-masterclass-fur-hair-grooming-toolset/>. Accessed: August 7, 2019.
- WEI, X. 2006. What is human hair, a light and scanning electron microscopy study. <http://www2.optics.rochester.edu/workgroups/cml/opt307/spr06/xue/project.htm>. Accessed: August 05, 2019.
- YAN, L.-Q., JENSEN, H. W., AND RAMAMOORTHI, R. 2017. An efficient and practical near and far field fur reflectance model. *ACM Trans. Graph.* 36:67:1–67:13.
- YAN, L.-Q., TSENG, C.-W., JENSEN, H. W., AND RAMAMOORTHI, R. 2015. Physically-accurate fur reflectance: Modeling, measurement and rendering. *ACM Trans. Graph.* 34:185:1–185:13.
- YAN, L.-Q., TSENG, C.-W., JENSEN, H. W., AND RAMAMOORTHI, R. 2016. Physically-accurate fur reflectance: Modeling, measurement and rendering. <https://cseweb.ucsd.edu/~viscomp/projects/fur/>. Accessed: August 9, 2019.
- YUKSEL, C. 2006. Hair model files. www.cemyuksel.com/research/hairemodels. Accessed: August 7, 2019.
- ZAAL, G. 2016. Schadowplatz (HDRI Haven). <https://hdrihaven.com/hdri/?h=schadowplatz>. Accessed: July 28, 2019.
- ZAAL, G. 2017a. Misty pines (HDRI Haven). https://hdrihaven.com/hdri/?h=misty_pines. Accessed: July 28, 2019.
- ZAAL, G. 2017b. Vatican road (HDRI Haven). https://hdrihaven.com/hdri/?h=vatican_road. Accessed: April 12, 2019.

ZINKE, A. AND WEBER, A. 2007. Light scattering from filaments. *IEEE Transactions on Visualization and Computer Graphics* 13:342–356.

ZINKE, A., YUKSEL, C., WEBER, A., AND KEYSER, J. 2008. Dual scattering approximation for fast multiple scattering in hair. *ACM Trans. Graph.* 27:32:1–32:10.

APPENDIX A

USER MANUALS

To add **PBR Hair Shader Extended** tool to Houdini, add folder with shader code and digital asset to HOUDINI_PATH environment variable.

Listing A.1: HOUDINI_PATH variable

```
HOUDINI_PATH = PATH_TO_FOLDER/houdini_configs ;&
```

This variable may be added to houdini.env file, which, depending on the system, may be found, for example in:

Listing A.2: Path to houdini.env file

```
/Users/username/Library/Preferences/houdini/17.5/
```

CyHairConvert tool may be used to convert any cyHair file with .hair format. To do it, in terminal has to be executed the following command:

Listing A.3: Convert from cyHair format

```
./cyHairConvert readHair.hair writeHair.txt
```

To upload converted cyHair file to houdini, just navigate to in File Parameter it in Hair Loader.



Figure A.1: Uploading converted cyHair file to Houdini