SAPIENZA
UNIVERSITÀ DI ROMA

Rendering human hair in Yocto/GL

Facoltà di Ingegneria dell'informazione, informatica e statistica
Artificial Intelligence and Robotics

Andrea Gervasio
ID number 1883259

Advisor
Prof. Fabio Pellacini

Academic Year 2022/2023

Rendering human hair in Yocto/GL
Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: August 26, 2023

Author's email: gervasio.1883259@studenti.uniroma1.it

Abstract

Although the Yocto/GL [1] renderer includes some primitive functions to render animal fur and human hair, these contain different approximations which make the final results not realistic.

In this report I will describe the work done to expand the Yocto/GL renderer in order to implement a hair scattering model, as described in the paper *The implementation of a hair scattering model* [2] by Matt Pharr. This will be done by implementing a BRDF and a new type of material. Figure 0.1 shows a model of curly hair obtained using the expanded renderer.

Figure 0.1. Blonde hair rendered using the scattering model described in this document.

Contents

Chapter 1

Introduction

1.1 Structure of the project

The changes are mainly focused in the *libs\yocto_hair* subdirectory. This choice was made for two main reasons:

- to minimize the changes made to the core structure of the renderer;
- to underline the fact that this is an extension, so the renderer could perfectly work without this implementation.

Having said that, there were some aspects of the code which needed to be included in some files of the original renderer. In particular:

- in the *libs\yocto\yocto_scene.{h, cpp}*, to add the new hair material and its attributes;
- in the *libs\yocto\yocto_sceneio.cpp*, to read the new attributes of the hair material which can appear in the json files describing the scenes to render;
- in the *libs\yocto\yocto_pbrtio.{h, cpp}*, to handle the conversion of the pbrt materials into the Yocto/GL ones, the new hair material has been added;
- in the *libs\yocto\yocto_trace.cpp*, to implement the behaviour of the path tracer when it encounters the new hair material in the scene.

1.2 Geometry

I used Benedikt Bitterli's rendering resources [3] to test my implementation, with the due changes in order to use them with the Yocto/GL renderer. I converted the *.pbrt* scenes in *.json* files. Since Yocto/GL does not support Bézier curves, I approximated them with two or four straight lines to render straight and curly hair respectively. All the lines belonging to the same model are stored in a *.ply* shape file.

It is important to note that, during the implementation of the scattering model, the scattering in the longitudinal plane and the one in the azimuthal plane are considered separately. In this paper, u is used to denote the direction along the

length of the curve, while v denotes its width. At a given u, all the possible surface normals of the curve are given by the surface normals of the cross-section at that point, and they all lie in a single plane, the *normal plane*.

The directions at a ray-curve intersection point will be represented with respect to coordinated (θ, ϕ) which are defined with respect to the normal plane at the u position where the ray intersected the curve. The θ angle is the longitudinal angle, which is the offset of the ray with respect to the normal plane, and it ranges from $-\pi/2$ to $\pi/2$. The ϕ angle is the azimuthal angle and it is found by projecting a direction ω into the normal plane and computing its angle with the y axis. It ranges from 0 to 2π. See Figure 1.1 for a graphical representation of the two angles.

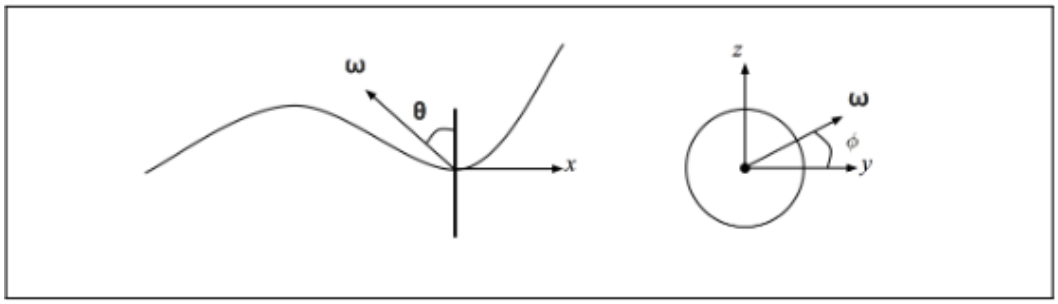

Figure 1.1. The definitions of the θ and ϕ angles.

Consider incident rays with some direction ω: at any given parametric u value, all such rays that intersect the curve can only possibly intersect one half of the circle swept along the curve.

The circle diameter will be parameterized with the variable h, with range $[-1, 1]$, which corresponds to the distance of the ray intersection from the center of the circle. In the code, h is computed as $-1 + 2v$. Given h for a particular circle intersection, the angle between the surface normal and the direction ω, denoted as γ is computed as $sin\gamma = h$.

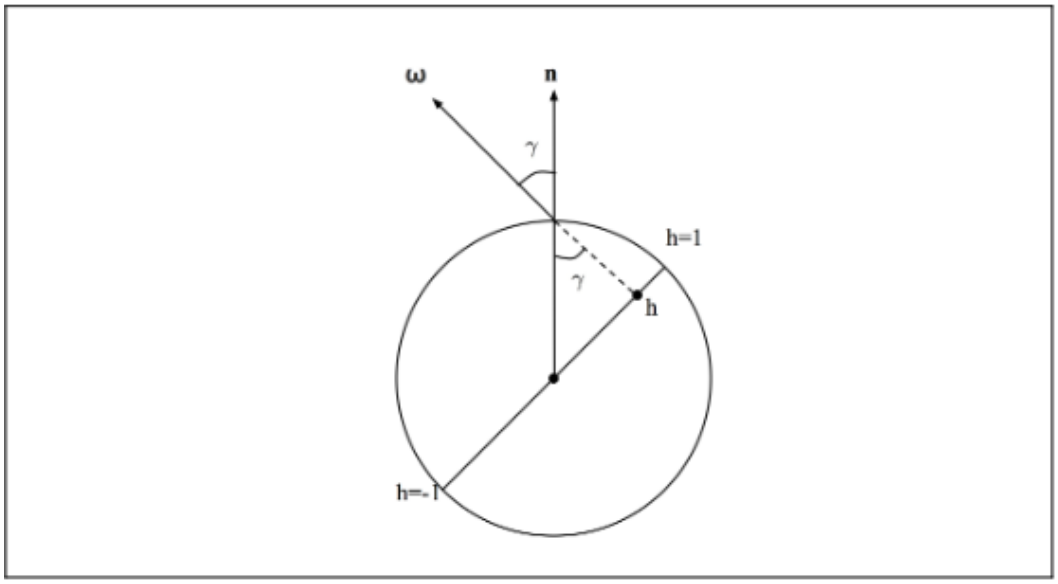

Figure 1.2. Graphical representations of h and the γ angle.

1.3 Utility Functions

The Pharr's paper and the pbrt code use some utility functions introduced to transcribe equations to code in a simpler way. It is also stated in the paper that using these (in particular the *pow* one) speeds up the code to a factor of 4.6x on a 2016-era laptop. I tried both with these functions and the ones implemented in Yocto/GL, which use the *std* functions of C++, but I did not see this speed up factor. In fact, the time needed to render an image was basically the same. Since the resulting images were also the same, I decided to use them anyway in the code in the eventuality that someone might have a machine older than the one I tested the code on and in the hope that the code is actually faster.

The functions implemented are:

- the *Sqr()* function, which computes the square of a given value;
- the *Pow()* function, which computes the power of a given value: this is done by splitting the exponent in half and recursively calling the function;
- the *SafeASin()* and the *SafeSqrt()* functions, which respectively compute the arcsine and square root of a given value by clamping it to the valid range, which the values computed during the computations could be slightly out of because of the floating-point round-off error.

Chapter 2

Scattering from hair

This chapter is dedicated to discuss the general scattering behaviours that give hair their distinctive appearance and the assumptions made in the development of the code.

2.1 Hair structure

Hair and fur have three main components:

- **Cuticle**: the outer layer. Its surface can be seen as a nested series of scales at a slight angle with respect to the hair surface;
- **Cortex**: the next layer inside the cuticle. It is typically colored with pigments that mostly absorb light;
- **Medulla**: the center core at the middle of the cortex. It is also pigmented. Scattering from the medulla is much more significant than scattering from the cortex.

Incident light arriving at a hair may be scattered more times before leaving the hair (Figure 2.1). To denote the number of path segments the light follows inside the hair before being scattered out we will use p. In the code p has a constant value of 3.

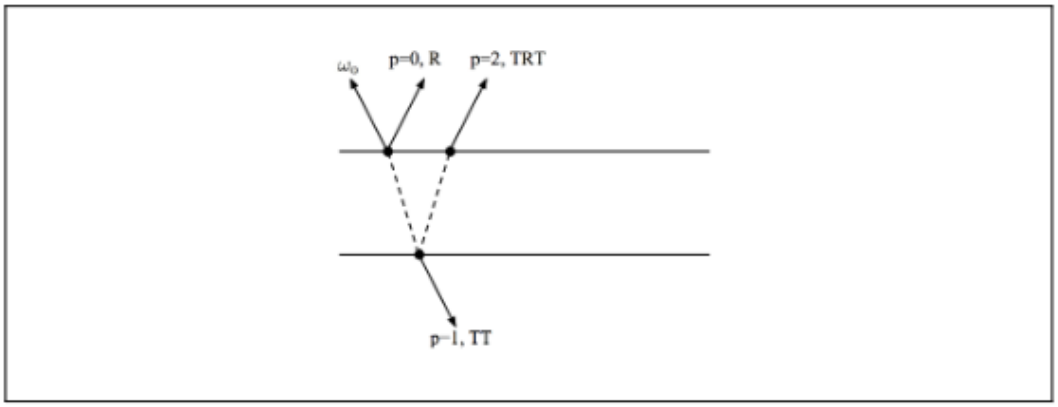

Figure 2.1. Number of segments the light makes before leaving the hair.

2.1.1 Model assumptions

The paper's author makes some assumptions about the hair geometry during the implementation:

- the cuticle will be modeled as a rough dielectric cylinder with scales that are all angled at the same angle α (Figure 2.2);
- the hair interior will be modeled as a homogeneous medium that only absorbs light, so that the scattering inside the hair will not be modeled directly;
- the scattering can be modeled accurately by a BRDF, with light entering and exiting the hair in the same place.

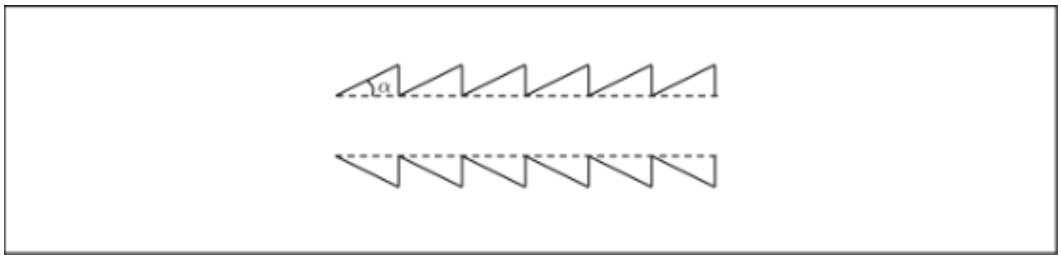

Figure 2.2. The hair is modeled as a nested series of cones. α is usually between 2°and 4°. It is bigger in the image for illustrative purposes.

2.1.2 The hair BRDF

The hair BRDF can be written as

$$f(\omega_o, \omega_i) = \sum_{p=0}^{\infty} f_p(\omega_o, \omega_i)$$

To make the scattering model implementation and importance sampling easier, a *semi-separable* model can be written as

$$f_p(\omega_o, \omega_i) = \frac{M_p(\theta_0, \theta_i) A_p(\omega_0) N_p(\phi)}{|cos\theta_i|}$$

Where M_p is the longitudinal scattering function, A_p is the attenuation function, and N_p is the azimuthal scattering function. The division by $|cos\theta_i|$ cancels out the corresponding factor in the reflection equation.

2.2 Longitudinal scattering

The longitudinal scattering is responsible for the specular lobe along the length of hair and the longitudinal roughness β_m controls the size of this highlight.

The model implemented was developed by d'Eon et al. [4]. The goal was to derive a scattering function that is normalized (ensuring both conservation and no energy loss) and can be sampled directly. The model is:

$$M_p(\theta_0, \theta_i) = \frac{1}{2vsin(1/v)} e^{-\frac{sin\theta_i sin\theta_0}{v}} I_0\left(\frac{cos\theta_0 cos\theta_i}{v}\right)$$

where I_0 is the modified Bessel function of the first kind and v is the roughness variance.

This model is not numerically stable for low roughness variance values, so a different approach was derived, which operates on the log of I_0 before taking an exponent at the end.

In the code this is achieved with a simple *if-else* statement.

The parameter of the hair material which controls the roughness is β_M. Figure 2.5 shows images rendered with different values.

2.2.1 Absorption in fibers

The A_p term describes how much of the incident light is affected by each of the scattering modes p. It embodies two effects:

- **Fresnel reflection and transmission** at the air-hair boundary;
- **absorption of light** that passes through the hair for $p > 0$. This is what gives hair and fur its color. (Figure 2.6)

The A_p function implemented in the code models all reflection and transmission at the hair boundary as perfectly specular (as opposed to M_p and N_p which model glossy reflection and transmission).

We will start by finding the transmittance of a single transmitted segment through the hair, in other words, we need to find the distance the ray travels until it exits the cylinder. The easiest way to do this is to compute the distances in the longitudinal and azimuthal projection separately.

To compute these distances we need the transmitted angles of the ray ω_0 in the longitudinal and azimuthal planes, which we will call θ_t and γ_t, respectively. Using Snell's law using the hair's index of refraction η allows us to compute $sin\theta_t$ and $cos\theta_t$.

$$sin\theta_t = \frac{sin\theta_0}{\eta}$$

$$cos\theta_t = \sqrt{1 - sin^2\theta_t}$$

Although we could compute the transmitted direction γ_t from γ_0 and project it into the normal plane, it's possible to compute γ_t directly using a modified index of refraction, which account for the effect of the longitudinal angle on the refracted direction in the normal plane. The modified index of refraction is given by:

$$\eta' = \frac{\sqrt{\eta^2 - sin^2\theta_0}}{cos\theta_0}$$

Figure 2.3 shows how, considering the azimuthal projection of the transmitted ray in the normal plane, the segment makes the same angle γ_t with the circle normal at both of its endpoints. Denoting the total length of the segment by l_a, $l_a/2 = cos\gamma_t$, assuming a unit radius circle.

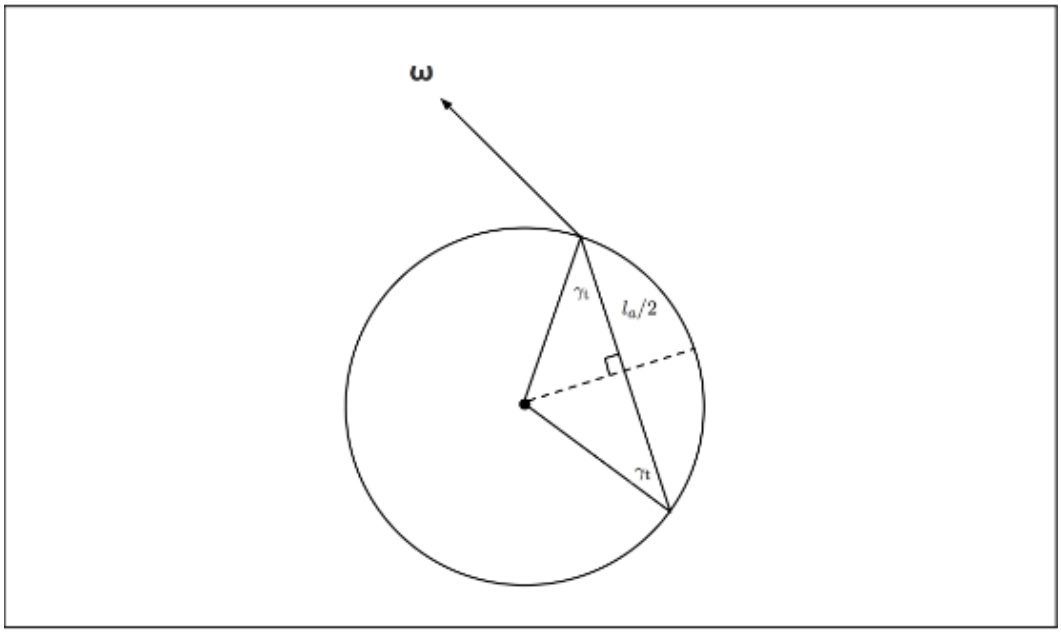

Figure 2.3. The distance of a transmitted ray.

Figure 2.4 shows how the distance that a transmitted ray travels before exiting is scaled by a factor of $1/cos\theta_t$ as it passes through the cylinder.

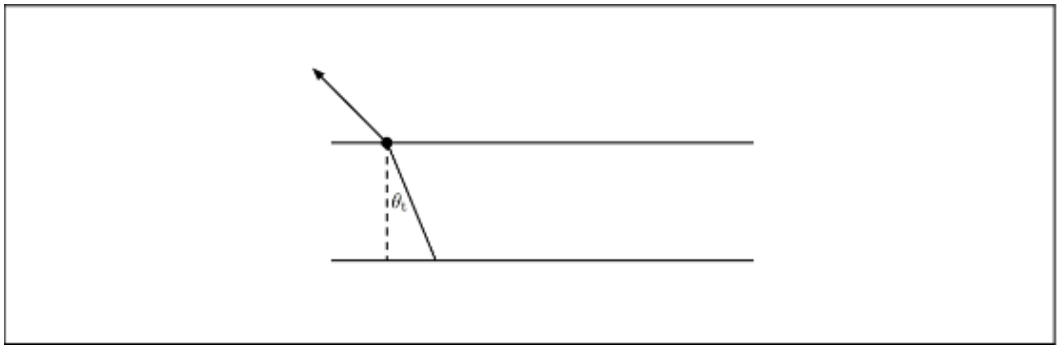

Figure 2.4. The scaling factor of a transmitted ray.

Putting these together, the total segment length with respect to the hair diameter is:

$$l = \frac{2cos\gamma_t}{cos\theta_t}$$

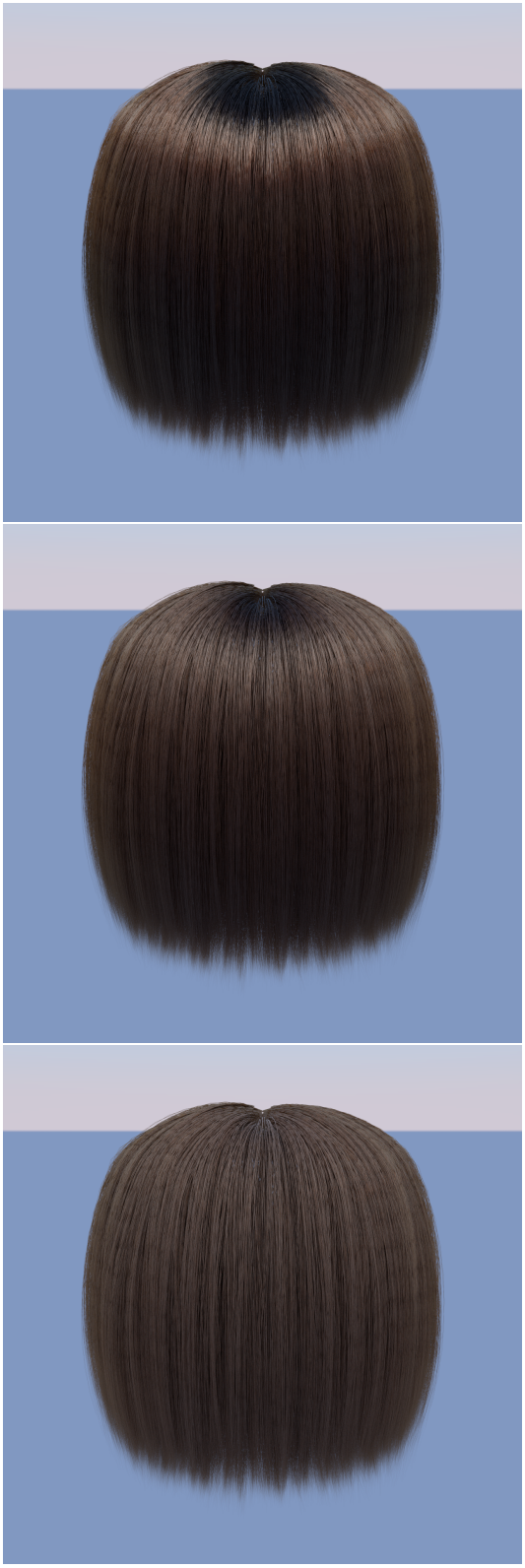

Figure 2.5. Hair model rendered with different values of β_M. The first uses a very low roughness, 0.1, and makes the hairs appear too shiny. The second one uses roughness 0.3 with highlight typical of human hair. The third one uses a very high roughness, 0.6, which makes the hair appear unrealistically flat.

Figure 2.6. Hair model rendered with different values of σ_A. The first uses a very low absorption, {0.06, 0.10, 0.20}, making the hair blonde because it reflect almost all the light. The second one uses absorption {0.84, 1.39, 2.74} making the hair a shade of brown. The third one uses a very high absorption, {3.35, 5.58, 10.96}, which absorbs almost all the light, making the hair appear black.

2.3 Azimuthal scattering

In this section we will discuss the component of scattering dependent on the angle ϕ. We will do this entirely in the normal plane.

The azimuthal scattering model works by computing a new azimuthal direction assuming perfect specular reflection and transmission and then by defining a distribution of directions around this central direction. The width of this distribution is directly proportional to the roughness.

Figure 2.7 shows how an incident ray is deflected by specular reflection and transmission in the normal plane for the first two values of p.

- $p = 0$: the incident and reflected directions make the same angle γ_0 with the surface normal. The new change in angle is thus $-2\gamma_0$;
- $p = 1$: the ray is deflected from γ_0 to γ_t when it enters the cylinder and then correspondingly on the way out. When the ray is transmitted again out of the circle, it again makes an angle γ_0 with the surface normal.

Adding up the angles, the net deflection is $2\gamma_t - 2\gamma_0 + \pi$.
The function to compute the net change in azimuthal direction is

$$\Phi(p, h) = 2\gamma_t - 2\gamma_0 + \pi$$

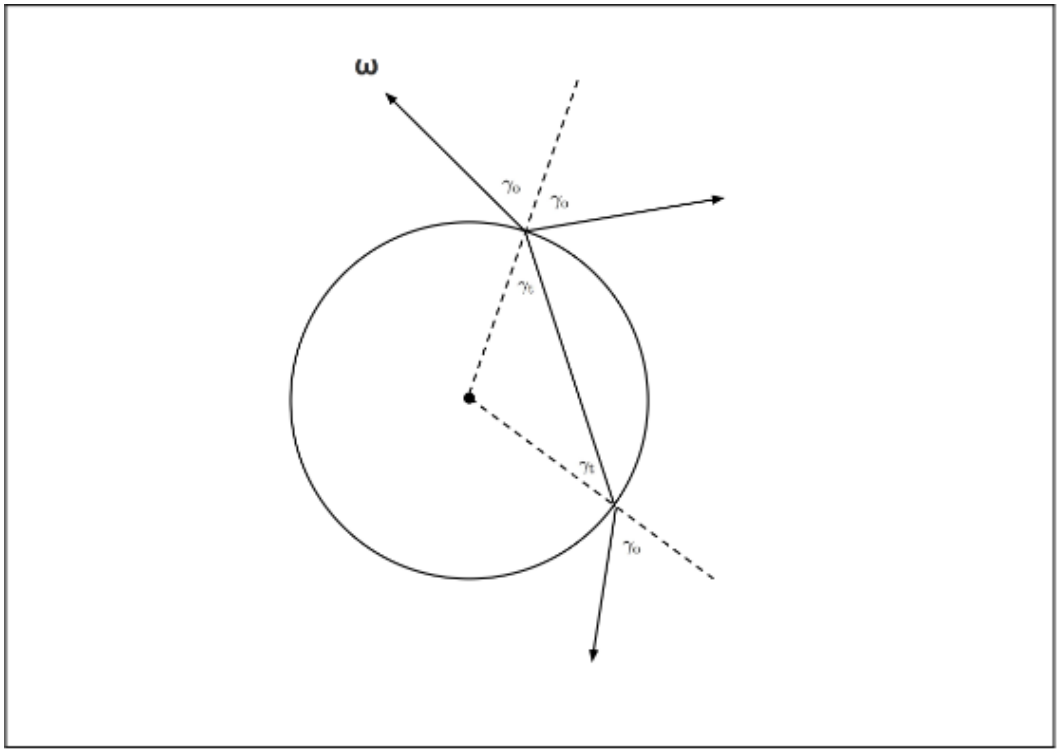

Figure 2.7. Deflection of an incident ray in the normal plane.

We now need a way to represent surface roughness, so that a range of direction centered around the specular direction can contribute to scattering. The logistic

distribution provides a good option, since it is similar in shape to the Gaussian, while also being normalized and integrable in closed-form.

The logistic distribution takes a scale factor s, which controls its width:

$$l(x, s) = \frac{e^{\frac{x}{s}}}{s(1 + e^{\frac{x}{s}})^2}$$

Because the logistic distribution is normalized, it is its own PDF. Its integral is reasonably straightforward:

$$\int l(x, s)\, dx = \frac{1}{1 + e^{\frac{-x}{s}}}$$

It is useful to define a normalized logistic function over a range [a, b], the trimmed logistic l_t. We will only use it in the range $[-\pi, \pi]$, but the general formula is:

$$l_t(x, s, [a, b]) = \frac{l(x, s)}{\int_a^b l(x', s)\, dx'}$$

The N_p function computes the angular difference between ϕ and $\Phi(p, h)$ and evaluates the azimuthal distribution with that angle.

This difference may be outside the range we've defined the logistic over, $[-\pi, \pi]$, so we rotate around the circle as needed to get the value to the right range. Because the difference never gets too far out of the range for the small p used here, we just add or subtract 2π as needed.

As with the longitudinal roughness, it's helpful to have a roughly perceptually linear mapping from azimuthal roughness $\beta_n \in [0, 1]$ to the logistic scale factor s.

$$s = \sqrt{\pi/8} * (0.256\beta_n + 1.194\beta_n^2 + 5.372\beta_n^{22})$$

Figure 2.8 shows rendering of a hair model with fixed longitudinal scattering and varying azimuthal scattering. The image shows how a high azimuthal roughness causes the hair to be lighter in color. This is because more light is able to exit the hair after multiple scattering when the distribution is broader.

Figure 2.8. Hair model rendered with different values of β_n, 0.3, 0.6 and 0.9 respectively.

Chapter 3

Implementation of the scattering model

This chapter is dedicated to discuss the implementation of the scattering model, focusing on the code and how it was developed to fit in the Yocto/GL renderer.

3.1 The hair material

To model the behaviour of hairs, I created a new structure *hair_material* with the following parameters:

- **sigmaA**: the absorption coefficient of the hair interior, where distance is measured with respect to the hair cylinder's diameter;
- **betaM**: the longitudinal roughness of the hair, mapped to the range [0, 1];
- **betaN**: the azimuthal roughness, also mapped to [0, 1];
- **alpha**: the angle that the small scales on the surface of hair are offset from the base cylinder, typically 2°.
- **eta**: the index of refraction of the interior of the hair, typically 1.55;
- **color**: the color of the hair;
- **eumelanin**: the eumelanin concentration in the hair;
- **pheomelanin**: the pheomelanin concentration in the hair

Eumelanin and pheomelanin are two pigments responsible for the hair color. The eumelanin concentration gives the most common hair colors (black, brown and blonde), while the pheomelanin concentration gives red and orange hair.

3.2 The hair BRDF

The hair BRDF implemented in the code is parameterized by four values:

- **eta**, **sigmaA** and **alpha** are the same as the *hair_material* structure;

- **h**: the [–1, 1] offset along the curve width where the ray intersected the hair;

As written in section 2.1.2, the hair BRDF can be written as

$$f(\omega_o, \omega_i) = \sum_{p=0}^{\infty} f_p(\omega_o, \omega_i)$$

and f_p can be written as

$$f_p(\omega_o, \omega_i) = \frac{M_p(\theta_0, \theta_i) A_p(\omega_0) N_p(\phi)}{|cos\theta_i|}$$

so, to evaluate the hair scattering model, we need the sine and cosine of θ that each direction makes with the plane perpendicular to the curve, and the angle ϕ in the azimuthal coordinate system.

Sin θ_0 is given by the x component of ω_0 in the BRDF coordinate system, as seen in section 1.2. Since $\theta_0 \in [-\pi/2, \pi/2]$, $cos\theta_0$ is positive, so we can compute it using the identity $sin^2\theta + cos^2\theta = 1$. The angle ϕ_0 in the perpendicular plane is computed with std::atan.

The same holds for θ_i and ϕ_i with respect to ω_i.

3.2.1 The implemented functions

To respect the Yocto/GL convention, I implemented three new functions:

- ***eval_hair_scattering***: evaluates the hair BRDF according to the input incoming and outgoing directions;
- ***sample_hair_scattering***: given an outgoing direction, samples an incoming direction according to the hair BRDF;
- ***sample_hair_scattering_pdf***: returns the pdf for the hair BRDF sampling related to the given incoming and outgoing directions.

To evaluate the input *hair_material* together with the v coordinate of the ray-line intersection and to build the BRDF, another function, ***eval_hair_brdf*** has been implemented. It also uses a frame to convert from world to local BRDF coordinate system and vice versa, since pbrt computations are made in the BRDF coordinate system, while Yocto/GL uses the world coordinate system. The z axis of this frame is oriented along the shading normal, while the x axis is oriented along the line tangent.

3.3 The importance sampling

At low roughnesses, the hair BRDF varies rapidly as a function of direction, so it is crucial for efficient rendering to be able to generate sampled directions and compute the PDF for sampling a given direction according to a distribution that is similar to the overall BRDF.

In the code, samples are generated using two steps:

- first, we choose a p term to sample according to a probability based on each term's A_p function value, which gives its contribution to the overall scattering function;

- second, we find a direction by sampling the corresponding M_p and N_p terms.

3.3.1 Computing additional sample values

Since a total of four random samples are needed to sample the direction ω_i, but we only have two, we will implement an approach that lets us extract two separate samples from each provided sample.

This will achieved with the function *DemuxFloat()*, which decomposes a sample $\xi \in [0, 1)$ into a pair of samples, while also making some effort to preserve stratification in the return sample value.

The *DemuxFloat()* function uses the Morton curve to achieve its goals. This is a space-filling 1D curve that maps real numbers in [0, 1] to n-dimensional numbers $[0, 1]^n$. Using a 1D sample value ξ as an offset into a Morton curve, we can use each of the dimensions' values as independent samples.

An important advantage of using a Morton curve for this task is that it preserves stratification in the sample values it returns. To see why this is so, consider a collection of four stratified 1D sample values (i.e., one in [0, 1/4), one in [1/4, 1/2), and so forth.) The first sample value will be mapped mapped to a point in [0, 1/2) $\times$ [0, 1/2) by the Morton curve, since the first quarter of the 2D Morton curve traces out that region of space in two dimensions. Similarly, the next stratified 1D sample will be in the range [1/2, 1) $\times$ [0, 1/2), and so on.

To help achieving this, we use the *Compact1By1()* function, which takes a 32-bit integer, deletes all the bits with odd indices, and compacts the remaining bits. [5]

We will treat the sample ξ as a fixed-point value v computed by multiplying by 2^{32}. The 2D Morton curve effectively takes alternating bits from this value, giving two values in $[0, 2^{16}]$. These are then mapped back to Floats.

3.3.2 A distribution for sampling P

Next, we need a discrete PDF with probabilities for sampling each term A_p according to its contribution relative to all of the A_p terms, given θ_0. This will be achieved by the *ComputeApPdf()* method, which starts by computing the values of A_p for $cos\theta_0$. Next, the spectral A_p values are converted to scalars using their luminance and these values are normalized to make a proper PDF.

3.3.3 Sampling incident directions

Since we only need to generate one sample from the PDF's distribution, the work to compute an explicit CDF array is not something we are interested in. Given the PDF over the A_p terms, it is sufficient to loop over PDF values until the first value of p is found such that the sum of preceding PDF values is greater than the sample value.

With this term, we can sample the corresponding M_p term given θ_0 to find θ_i.

After sampling direction θ_i, we apply the inverse of the rotation that will later be used to account for hair scales when the BRDF is evaluated.

Next we will sample the azimuthal distribution N_p. For terms up to $pMax$, we take a sample from the logistic distribution centered around the exit direction, which is given by $\Phi(p, h)$. For the last term, we sample from a uniform distribution.

By inverting the CDF of the trimmed logistic, we can derive the recipe to sample from its distribution given a random variable $\xi \in [0, 1)$:

$$\xi = \int_a^x l_t(x', s, [a, b])\, dx'$$

$$= \frac{1}{\int_a^b l(x, s)\, dx} \int_a^x l(x', s)\, dx$$

$$\frac{1}{\int_a^b l(x, s)\, dx}\left(\frac{1}{1 + e^{\frac{-x}{s}}} - \frac{1}{1 + e^{\frac{-a}{s}}}\right)$$

With some algebra, it can be solved for x:

$$x = -slog\left(\frac{1}{\xi \int_a^b l(x, s)\, dx + \frac{1}{1+e^{\frac{-a}{s}}}} - 1\right)$$

In practice, due to floating-point round-off, the implementation may compute an infinite value when $\xi \to 1$; a clamp at the end ensures that the returned value is in the range [a, b].

Given θ_i and ϕ_i, we can compute the sampled direction ω_i. Because we could sample directly from the M_p and N_p distributions, the overall PDF is:

$$\int_{p=0}^{pMax} M_p(\theta_0, \theta_i)\tilde{A}_p(\omega_0)N_p(\phi)$$

where $\tilde{A}_p$ are the normalize luminance-weighted PDF terms. Note that θ_i must be shifted to account for hair scales when evaluating the PDF.

3.4 The implemented tests

To check the physical correctness of the code, the author of the paper implemented some tests:

- ***The White Furnace Test***: checks if, when the hair does not have any absorption ($\sigma_A = 0$), all the incident light gets reflected;
- ***The White Furnace Sampled Test***: same as above, but uses sampled incident directions, instead of a uniform one.
- ***The Sampling Weights Test***: checks if the ratio between the BRDF value and the PDF computed for the direction ω_i is 1, as long as there is no absorption in the hair;
- ***The Sampling Consistency Test***: checks if the importance sampling scheme implemented has the same result as using a uniform distribution of directions over the unit sphere.

The tests are called in the *path_trace* function in the *yocto/yocto_trace.cpp* file, but they are commented to speed up the computations.

3.4.1 The White Furnace Test

The White Furnace Test checks if all the light gets reflected by the hair if it does not have any absorption when it is illuminated with uniform incident radiance.

If that is the case, the reflected radiance should be the same as the incident radiance.

The implementation tests a variety of azimuthal and longitudinal roughnesses. For each roughness, we compute a Monte Carlo estimate of the spherical–directional reflectance. Each sample is evaluated by first sampling a random offset along the hair h and then computing the fraction of reflected radiance for a random incident direction.

3.4.2 The White Furnace Sampled Test

The White Furnace Sampled Test sample incident directions rather than using a uniform distribution and then divides the BRDF values by the PDF to compute the estimate of the reflectance.

It uses a tighter tolerance ($1 \pm .01$) than the original White Furnace Test, since importance sampling gives much faster convergence than uniform sampling over the sphere.

3.4.3 The Sampling Weights Test

Because the sampling routine implemented in the code matches the PDF of the underlying BRDF, the Sampling Weight Test checks if the ratio between the BRDF value and the PDF computed is 1.

Performing the test is mostly a matter of setting up enough context to build the PDF and then verifying the ratio between the two quantities.

The code accepts a small amount of error (1 ± 0.001) in order to allow floating-point round-off error.

3.4.4 The Sampling Consistency Test

The Sampling Consistency Test checks if, given a sufficient number of samples, the result obtained using the custom importance sampling scheme implemented is the same as the one using a uniform distribution of directions over the unit sphere.

This is achieved by computing the reflected radiance from a varying incident radiance function with the scattering equation.

The L_i lambda function defines an incident radiance function with some variation as a function of direction. This function is kept fairly simple so that sampling the BRDF alone works well to compute reflected radiance and we can avoid the complexity of implementing multiple importance sampling in the test.

For each sample in the Monte Carlo estimate, we choose a random point on the hair and use a pair of random numbers to sample an incident direction. We then

use the regular Monte Carlo estimator to compute estimates using both sampling approaches.

The error allowed in this test is 5%. It is fairly high, but it was chosen so that the test can run quickly. With a higher number of samples, a close agreement could be expected, but in this way, the test runs in a second or so.

Chapter 4

Results

This chapter will show the results obtained by the model, using different scenes and different parameterizations. All the images are obtained using 1536 samples on a machine with a Intel i5-12600K CPU and a NVIDIA GeForce 3060Ti GPU, and Windows 10 as OS.

4.1 Scenes used

I tested the model using Benedikt Bitterli's rendering resources, after making the right correction to them to make them usable by the Yocto/GL renderer. The scenes are:

- a scene with four hair locks, all of different colors;
- a sphere with straight hairs attached to it;
- a sphere with curly hairs attached to it.

4.1.1 Hair locks

The scene is composed of four hair locks with difference colors, from left to right: black, red, brown and blonde. The hair color is controlled by the eumelanin and pheomelanin concentration (See section 4.2.1 for more information).

4.1.2 Straight hair

The scene is composed of a sphere with attached straight hair.

4.1.3 Curly hair

The scene is composed of a sphere with attached curly hair.

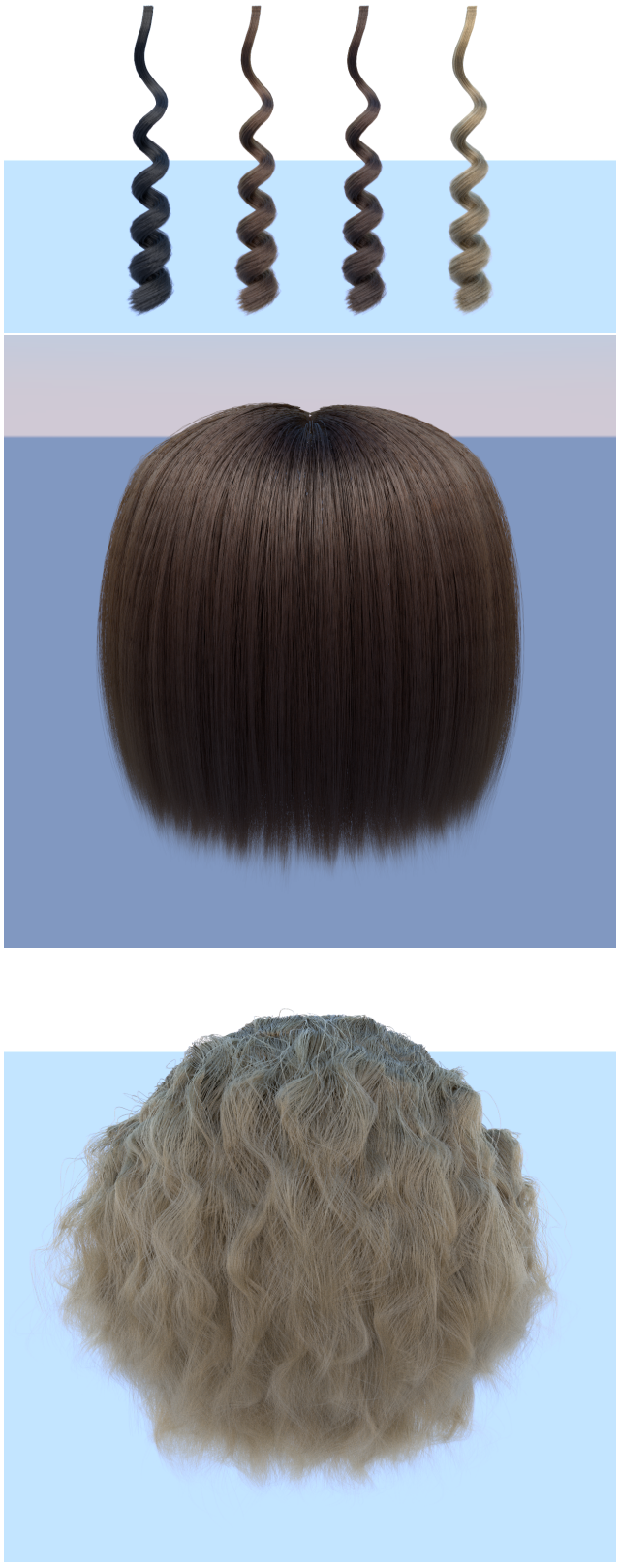

Figure 4.1. The scenes used by the renderer.

4.2 Parameters tested

I used different parameterizations to render the scene to see how the final results would change. In particular I tested:

- ***The eumelanin concentration*** to change the hair color;
- ***The absorption*** of the hair to change its color;
- ***The scale angle***;
- ***The longitudinal roughness*** to see the difference in the specular lobe along the length of hair;
- ***The azimuthal roughness*** to see the difference in light that exits the hair after the scattering.

4.2.1 Eumelanin tests

The eumelanin concentration in the hair changes its color. A value of 0.3 corresponds to blonde hair, 1.3 corresponds to brown hair, while a value of 8 results in black hair.

4.2.2 SigmaA tests

The absorption of the hair fibers is another method of defining the hair color, because it corresponds to the quantity of light that get absorbed by the them. A higher value is associated with a darker color of the hair.

4.2.3 Alpha tests

The angle of the scales in the hair is responsible for the secondary hair-colored highlight below the white highlight.

4.2.4 BetaM tests

The longitudinal roughness is responsible for the size of the specular lobe along the length of hair. A higher value corresponds to a higher dimension of the lobe, which can be unrealistic for human hair.

4.2.5 BetaN tests

The azimuthal scattering corresponds to the width of the azimuthal distribution, in other words, the quantity of light that is able to exit the hair after multiple scattering. A higher value causes the hair to be lighter in color.

Figure 4.2. The eumelanin tests. The image are rendered with a value of 8, 1.3 and 0.3 respectively.

Figure 4.3. The σ_A tests. The image are rendered with a value of {3.35, 5.58, 10.96}, {0.84, 1.39, 2.74} and {0.06, 0.10, 0.20} respectively.

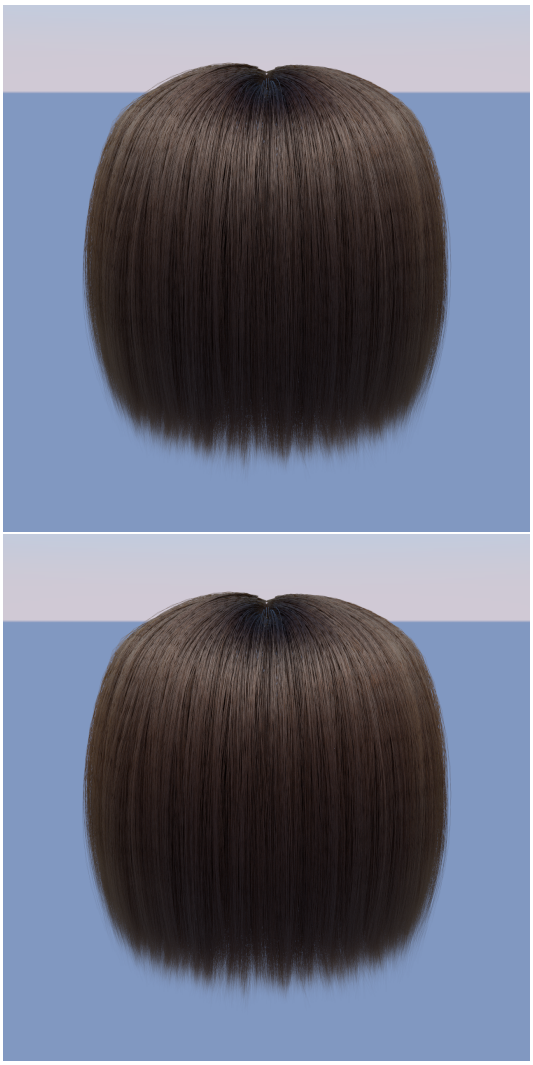

Figure 4.4. The α tests. The image are rendered with a value of 0°and 2°respectively.

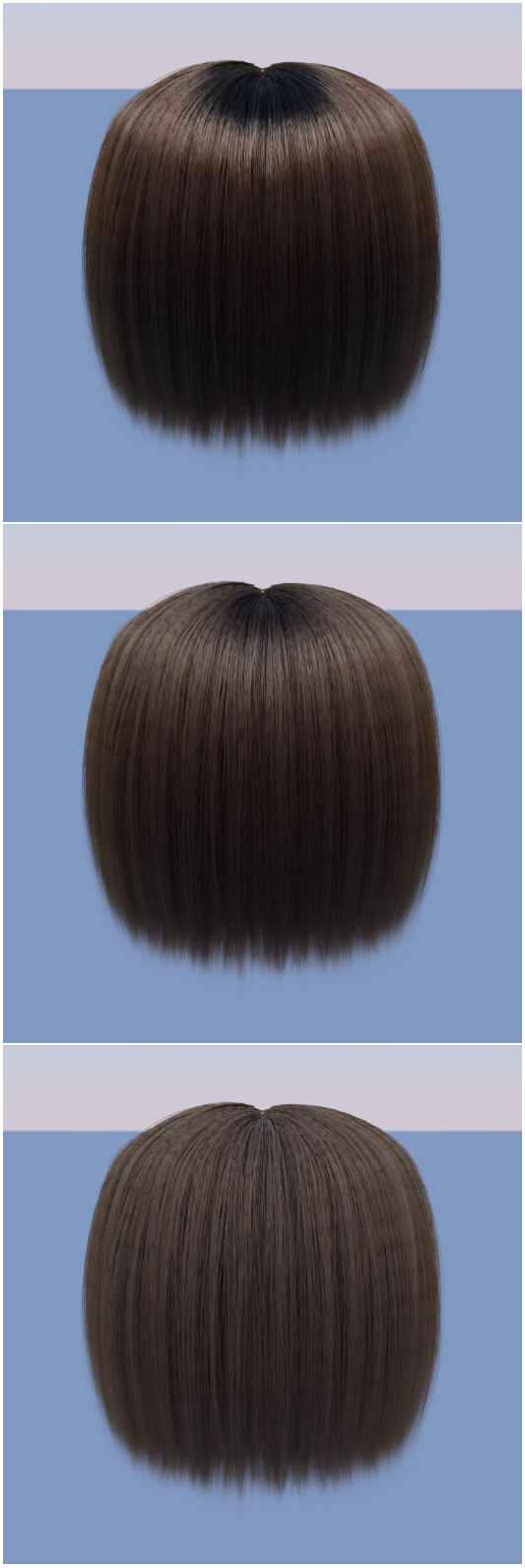

Figure 4.5. The β_M tests. The image are rendered with a value of 0.1, 0.3 and 0.6 respectively.

Figure 4.6. The β_N tests. The image are rendered with a value of 0.3, 0.6 and 0.9 respectively.

Bibliography

[1] Fabio Pellacini, "Yocto/gl: Tiny c++ libraries for data-oriented physically-based graphics," 2023.

[2] Matt Pharr, "The implementation of a hair scattering model," 2016.

[3] B. Bitterli, "Rendering resources," 2016. https://benedikt-bitterli.me/resources/.

[4] E. d'Eol, "An energy-conserving hair reflectance model," 2011. http://www.eugenedeon.com/wp-content/uploads/2014/04/egsrhair.pdf.

[5] F. Giesen, "Decoding morton codes," 2009. https://fgiesen.wordpress.com/2009/12/13/decoding-morton-codes/.