REAL-TIME STRAND-BASED HAIR RENDERING USING GUIDE HAIR INTERPOLATION AND DEPTH PEELING

PERSEMBAHAN HELAIAN RAMBUT WAKTU NYATA MENGGUNAKAN INTERPOLASI RAMBUT PANDU DAN PENGUPASAN KEDALAMAN

Jonathan Eu Jin Liew
Program of Mathematics with Computer Graphics,
Faculty of Science and Natural Resources,
Universiti Malaysia Sabah,
Kota Kinabalu, Sabah, Malaysia.
jonathan_liew_bs20@iluv.ums.edu.my

Rechard Lee
Mathematics Visualization Research Group (MathVis),
Faculty of Science and Natural Resources,
Universiti Malaysia Sabah,
Kota Kinabalu, Sabah, Malaysia.
rechard@ums.edu.my

Abstract— Hair is an important element in CGI and digital animation. However, it has been proven to be challenging to simulate hair properties when rendering hair. Lighting model that accounts for more than one light scattering direction renders high quality hair strands in terms of visual appearance at the cost of performance, while overlapping hair strands produce incorrect occlusion when accounting for hair transparency. The aim of this project is to render hair geometry in real-time and adapt order-independent transparency on hair strands. The approach used in this project involves rendering explicit hair strands using guide hair interpolation and adapting order-independent transparency for correct occlusion of hair strands using depth peeling.

Keywords—3D computer graphics; hair rendering; hair; light scattering; reflectance modeling; tessellation

I. Introduction

Hair is an important element in computer-generated imagery (CGI) and digital animation. It is used to express a fictional character's personality and feelings. The need for realistic computer graphics simulations and renderings has expanded dramatically as technology advances. As virtual, augmented, and extended reality become more widespread, demand for high-quality hair renderings rises. Hair is one of the more complex parts of a mammal. Thus, digital artists and back-end programmers struggle with hair modelling. Researchers and developers have been striving to find affordable and attractive ways to depict hair models since mammals have up to tens or hundreds of thousands of strands [1].

Hair strands have complex properties. Those properties are transparent, thin, and self-shadowing [1][2]. These properties are important when simulating light interaction on hair fibres to render physical-based hair models. However, the cost when accounting for all these properties become expensive and hard to be implemented in real-time applications. For example, simulating light scattering on hair fibres using Marschner bidirectional model [3] can provide a more physically accurate hair model but it takes a long time to render [4]. Alpha blending, a common rendering technique for transparency, is used to create translucent hair strands. However, overlapping hair strands make the sorting process impractical to be implemented [2][5].

This paper proposed a technique to render hair strands in real-time and adapt order-independent transparency on hair strands to resolve these issues. Explicit Marschner hair strands will be modelled and rendered using guide hair interpolation. Depth peeling is also implemented in the proposed model to adapt order-independent transparency for correct occlusion when rendering hair strands in the scene.

Section II reviews past work on real-time hair rendering applications and hair transparency. Section III discusses the methodology used in this project. All experimental setups used for the testing phase are discussed in Section IV and the results are shown and discussed in Section V. Lastly, this paper is concluded in Section VI.

II. Literature Review

Multiple hair rendering frameworks have been introduced in the field of real-time computer graphics. Most of these techniques are explicit or strand-based such as in [6] and include features such as light scattering, self-shadowing, and transparency, which are important for realistic or physically accurate hair renders [2]. Example of implicit hair representation is [7] where author constructed the hair volume using a set of guide hairs. Author chooses a subset of strands to be simulated and rendered. The selected subset is denoted as "guides" as they act as a reference to direct the pipeline in generating additional hair using interpolation to cover the remaining volume at a later stage.

There are two popular local illumination models that have been used in recent hair-rendering models such as in [8], which are the Kajiya-Kay model [9], and the Marschner model [3]. Kajiya-Kay model have been implemented in real-time applications due to its simplicity as the model only accounts for one scattered light ray on the surface whereas the Marschner model accounts for three components, the primary and secondary specular component, and the component where light completely transmits the hair fibre. However, studies such as in [10] has shown that Marschner model produced a more physically accurate hair model to real hair strands than the Kajiya-Kay model but renders at a higher cost when accounting for bidirectional light scattering.

Alpha blending is widely employed in hair-rendering for two purposes [2]. Translucent materials, such as hair, are often rendered using alpha blending. Second, hair strands are of sub-pixel size under normal viewing conditions and screen resolutions. Alpha blending is employed as an anti-aliasing technique to remove jagged edges at the tip of the strands. However, studies in [2] and [5] have shown geometry sorting by depth along the view direction is not well-suited for hair rendering as hair strands are highly overlapping, thus making it hard for all the strands to be sorted correctly before blending them together. This motivated researchers to come up with an alternative for alpha blending.

Order-independent transparency was introduced later. A depth-peeling method is implemented to ensure all pieces are pulled back-to-front [11]. The geometry peeled in layers in repetition with using depth tests to pass the farthest piece, using the depth buffer from the previous pass as a qualifier. Fragments are removed one layer at a time until an occlusion query indicates no more drawing. Authors in [12] have suggested there is only very little visual improvement to the final image after the second pass. There have been studies that have shown that multi-pass rendering is expensive and inefficient for real-time applications [13][15]. Researchers have come up with approximation methods such as [14] and [15]. However, these single-pass techniques require dynamic memory allocation to overcome challenges of running out of memory when storing the fragments in real time, where these techniques are still difficult to be implemented, even in modern GPUs [16].

III. Methodology

The system architecture (Fig. 1) of the proposed system is divided into five main components, which are input, asset loading, guide hair placement, shader programs, and depth peeling.

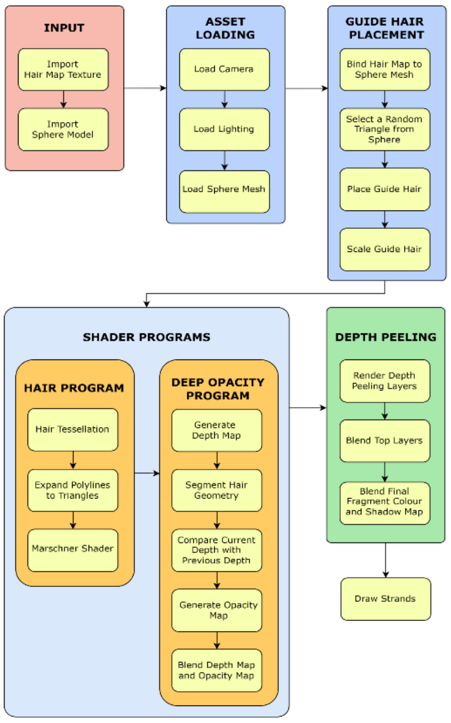

Fig. 1. Overview of system architecture.

A. *Input and Asset Loading*

The system imports a hair map texture image and a three-dimensional (3D) OBJ-format sphere model. These dataset inputs are obtained under MIT License. These datasets can be retrieved from a GitHub repository published by Emma O'Reilly in [17]. A lighting system and the imported sphere model are loaded into the scene along with a perspective camera looking at the sphere model.

B. *Guide Hair Placement*

Since hair strands inherit almost similar properties to one another, [13] used guide hairs to model a head of hair. Only guide hairs are fully simulated and the rest of the strands within the guide hair's neighbourhood can be interpolated.

The imported hair map texture is used to determine the placement of guide hair strands on the sphere model. Once the texture is mapped onto the sphere, a random triangle on the sphere is selected to place the generated guide hair. A random point within the triangle is chosen to place the guide hair is shown in Fig. 2. The alpha values are then fetched from the imported hair map's texture coordinates. If the alpha value of the pixel colour is not zero, a hair is placed, and the alpha value is used to scale its length.

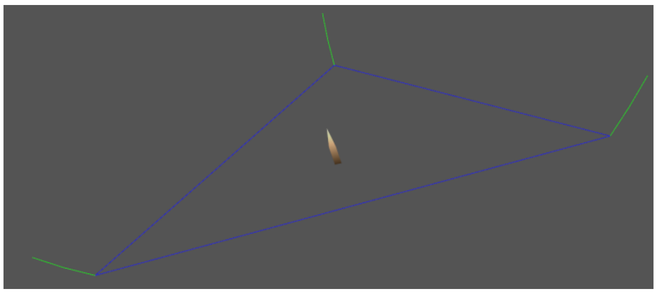

Fig. 2. Placement of guide hair in triangles on mesh.

C. *Programs*

Program objects are compilations of shaders. In this project, two shader programs will be used to add explicit details, consisting of Hair Program and Deep Opacity Program.

Tessellation is applied to the hair lines to generate more hair strands through interpolation using a single guide hair. More hair strands can be generated through a single GPU pass using this technique as shown in Fig. 3.

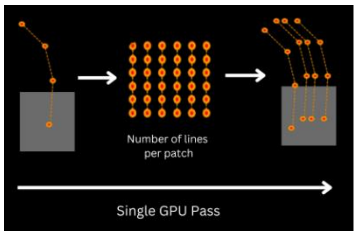

Fig. 3. Illustration of guide hair interpolation.

This technique begins by generating guide hair patches of the guide hairs. The hair strands are generated using Catmull-Rom spline interpolation. The start and end points along with their respective tangents are required to compute the curve, $f(t)$. Tangents at any point iteration, i is obtained using the previous and next vertex iteration as shown in (1), where the tightness of hair strand, a is set to 0.5.

$$T_i = a(P_{i+1} - P_{i-1}) \tag{1}$$

The generated line segments generated from the tessellation process are then expanded into triangles and shaded using Marschner lighting model. An illustration of the model is as shown in Fig. 4. Only local illumination is involved as the cost for global illumination application is too high to be implemented in real time [4].

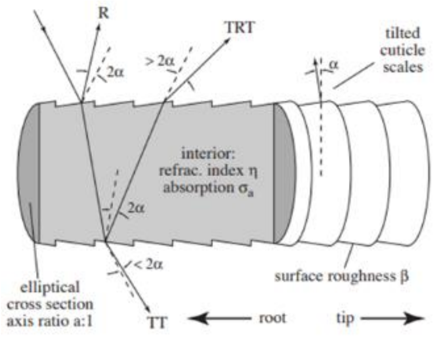
R
TRT
tilted cuticle scales
2α
>2α
α
interior: refrac. index η absorption σa
2α
<2α
surface roughness β
elliptical cross section axis ratio a:1
TT
root
tip

Fig. 4. Marschner's lighting model.

The ambient component computes the effect of hair-hair interaction when the model is not directly exposed to light.

The specular component, $\Psi_{specular}$, of the Marschner shader consists of three lighting components, R, TRT, and TT, where R stands for reflection and T stands for Transmission. R component is the primary specular peak where light is reflected off the hair fibre as soon as the ray touches the surface. TRT component is the secondary specular peak, where this component undergoes internal reflection. The TT component is the specular component to compute the surface colour when light ray completely transmits the hair fibre. These components are dependent on the angle of tilted cuticle scales, α, which is responsible for the shifting of glints and the surface roughness, β, that increases the width of glints. All the specular components are then added together.

The diffuse lighting component is responsible for the light that scatters throughout the hair and reflects in all directions. The diffuse reflection is determined by the dot product between the hair normal, $\boldsymbol{n}$ and the incident light vector, $\boldsymbol{l}$.

The final rendering equation, Ψ is obtained by computing the sum of the three components as shown below.

$$\Psi_R = \cos(\theta - \alpha_R)^{\beta_R} \tag{2}$$

$$\Psi_{TRT} = \cos(\theta - \alpha_{TRT})^{\beta_{TRT}} \tag{3}$$

$$\Psi_{TT} = \cos\emptyset \cos(\theta - \alpha_{TT})^{\beta_{TT}} \tag{4}$$

$$\Psi = k_a(C) + k_d(\boldsymbol{l} \cdot \boldsymbol{n}) + k_s(\Psi_R + \Psi_{TRT} + \Psi_{TT}) \tag{5}$$

where:

C is the base colour of hair strand.

$\emptyset$ is the rotation of hair strand around its axis.

The parameter values of longitudinal shifts and longitudinal widths for each lobe are set using the values in Table I where $\alpha_R = 5°$ and $\beta_R = 8°$. The lighting component intensities are set at $k_a = 0.5, k_d = 0.5$ and $k_s = 0.2$.

TABLE I. LIST OF DEFINED VALUES OF α AND β

Variable	Definition
Longitudinal shift of path R, α_R	[5,10]
Longitudinal shift of path TRT, α_{TRT}	$\frac{\alpha_R}{2}$
Longitudinal shift of path TT, α_{TT}	$\frac{3\alpha_R}{2}$
Longitudinal width of path R, β_R	[5,10]
Longitudinal width of path TRT, β_{TRT}	$\frac{\beta_R}{2}$
Longitudinal width of path TT, β_{TT}	$2\beta_R$

Hair shadowing is computed using Deep Opacity Maps as illustrated . Two render passes are required. First, hair shadow maps from the point of view from the light source are generated using depth buffers, where Z_0 is the depth value at which the hair geometry starts. The hair geometry is segmented into four layers, with each layer increasing in size from the light source.

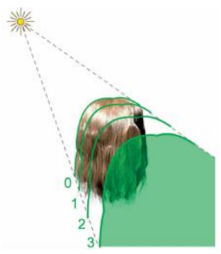

Fig. 5. Illustration of deep opacity maps.

Next, the deep opacity program calculates the values for opacity of each pixel on the image. Each colour channel of the opacity map is represented by a different layer of the shadow map. These layers influence the intensity values for red, green, blue, and alpha values of a pixel, where red is the first layer while alpha is the last. The pixels that fall in the last layer of the head geometry will have the darkest shadow as shown in Fig. 5. This step begins by obtaining Z_0 by performing a texture lookup on the shadow map and the depth values within each layer are determined. The value of layer size is set to 0.0005. The current depth, Z_i is then compared to Z_0 to determine which layer the pixel belongs. Once the opacities for all the pixels are obtained, the cumulative opacity of a layer at a certain pixel is determined by adding together the transparency values of all the fragments that contribute to that pixel and the opacity map is obtained. The shadow map and opacity map are then blended using additive blending to obtain the final shadow map.

D. Depth Peeling

Depth peeling is employed to simulate hair transparency. This technique involves three rendering passes to render the hair model in layers. This technique also involves two depth buffers. In each pass, the current pixel's depth is compared to the stored depth from the previous pass, and if it is closer to the camera, its colour is remained, else, the pixel colour is discarded. This is to render the hair fragments in layers from front to back. The layers are then alpha blended to create the final image. Blending the layers allow hair to look translucent and acts as an antialiasing technique. The result of alpha blending equation, $\bar{C}_{result}$ is shown below.

$$\bar{C}_{result} = \alpha_{source}\bar{C}_{source} + (1 - \alpha_{source})\bar{C}_{destination} \tag{6}$$

where:

$\bar{C}_{source}$ is the final colour of fragment shader.

$\bar{C}_{destination}$ is the colour vector already stored in frame buffer.

α_{source} is the alpha value of source colour vector.

IV. Experimental Setup

The default parameters used are as stated in Table II. Some parameters will be modified for performance comparisons in Section V, such as different noise amplitude and maximum hair length values for different hair type setups. The scene is lit up using only one key light.

TABLE II. Parameters used for testing phase.

Parameters	Value
Light intensity	1.5
Opacity, α_s	0.7
Total hair strands	25600
Total guide hairs	400
Strands per guide patch	64
Maximum hair length	0.5
Hair radius	0.001
Group spread	0.3
Hair colour	(97, 87, 77)
Noise amplitude	0.2
Noise frequency	0.5
Ambient intensity, k_a	0.5
Specular intensity, k_s	0.2
Phong specular exponent, p	7
Diffuse intensity, k_d	0.5
Longitudinal shift of R component, α_R	5
Longitudinal width, β_R	8
Light Colour	(255,255,255)
Light Intensity	1.5
Light Position	(-1.831, 7.465, 9.726)

During performance testing, different hair types are used to add stress to the proposed model. The hair types along with their condition parameters are as stated below, where the default hair type is normal hair.

i. **Normal hair**
Maximum hair length is set to 0.5 and noise amplitude is set to 0.1.

ii. **Curly hair**
Maximum hair length is set to 0.5 and noise amplitude is set to 0.5.

iii. **Long hair**
Maximum hair length is set to 1.0 and noise amplitude is set to 0.1.

iv. **Long and curly hair**
Maximum hair length is set to 1.0 and noise amplitude is set to 0.5.

The timing measurements have been captured with Nvidia's Nsight Graphics tool on an experimenting device that runs on NVIDIA GeForce GTX 1650 and Microsoft Windows 10 as its operating system. The captures have been done in the proposed system's editor with a scene viewport resolution of 1920x1080.

V. Results and discussion

This section discusses the results obtained from the testing phase of the proposed system. This section has two subtopics. The first subtopic discusses the hair rendering performance while the second subtopic discusses the visual result of depth peeling.

All performance tests were repeated for five iterations. All tests are done for 60 seconds as a constant variable to obtain the average frame rate and frame time for each iteration. The ideal and minimum benchmarking values are set to the general performance requirement for real-time simulations or applications as shown in Table III.

TABLE III. Performance benchmark values.

Requirement	Frame rate (fps)	Frame time (ms)
Minimum	30	33.33
Ideal	60	16.67

A. *Performance of Marschner Shader*

The performance testing for this section involves testing the influence of hair interpolation on the hair rendering performance. Tests involve different number of guide hair strands with a constant total number of hair strands rendered, different hair type setups with constant total number of hair strands, and different total number of hair strands with a constant number of strands per guide patch. Tests are done with self-shadowing enabled.

The first performance test is carried out on hairball simulation in 60 seconds using 400, 800, 1600 and 3200 guide hair strands respectively, where total hair strands are kept constant at 25600 hairs.

TABLE IV. Average Performance of Simulation Under Different Total Number of Guide Hair Strands used in 60 seconds.

Total Number of Guide Hairs	Frame Rate (fps)	Frame Time (ms)
400	56.02	17.85
800	30.30	33.00
1600	15.70	63.69
3200	8.10	123.46

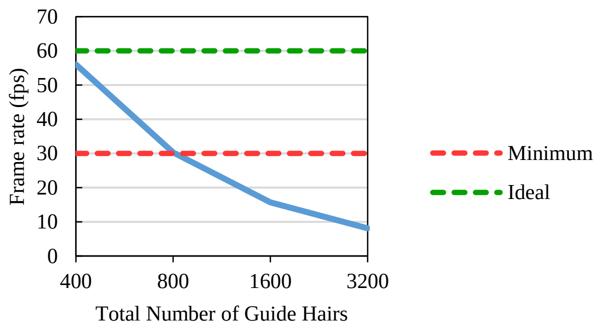

Fig. 6. Rendering performance under increasing number of guide hairs used. Total number of hair strands are kept constant at 25600 strands.

The influence of hair interpolation on the rendering performance can be seen in the results of Table IV and Fig. 6. 400 guide hairs used produced the highest performance of 56.02 fps and 17.85 ms while 3200 guide hairs produced the lowest performance of 8.10 fps and 123.46 ms. The results show that the frame rate decreases and frame time increases as the total number of guide hairs used increases. This is because as more guide hairs are used when total hairs are kept constant, more hair strands are required to be simulated and less hairs are interpolated within each guide patch, thus requiring more GPU passes. This leads to an increase in computational time and decrease in frame rate. From the graph above, the performance of shader drops below minimum real-time performance after the 800 guide hairs (32 hair strands per guide patch) mark, where 800 guide hairs recorded performance of 30.30 fps and 33.00 ms.

The next test is done using different hair type setups, normal, curly, long, and long and curly hair strands. This test is done to test the performance of Marschner shader on different hair types. These setups act as stress on the proposed system.

TABLE V. AVERAGE FRAME RATE AND FRAME TIME OF SIMULATION UNDER DIFFERENT HAIR TYPE SETUPS IN 60 SECONDS.

Hair Type	Frame Rate (fps)	Frame Time (ms)
Normal	56.02	17.85
Curly	49.14	20.35
Long	53.68	18.63
Long and Curly	44.58	22.43

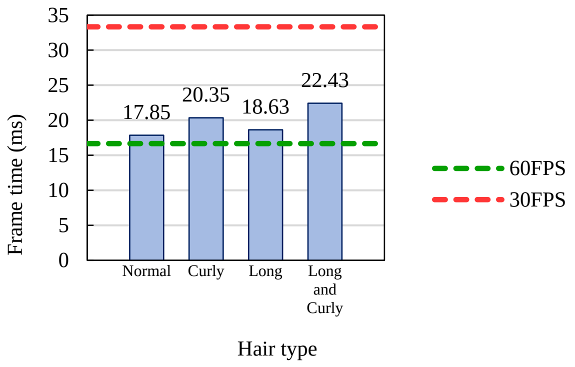

Fig. 7. Average frame times of different hair type setups.

From the values in Table V and Fig. 7, all hair type setups achieved real-time performance. Normal hair achieved highest frame rate of 56.02 fps and long and curly hair achieved the lowest frame rate of 44.58 fps. In terms of computation time, normal hair type renders the fastest at an average of 17.85 ms while long and curly hair renders the slowest at an average of 22.43 ms between frames. Rendering times for the curly hair and long hair setups are longer than that of normal hair (20.35 ms and 18.63 ms respectively). Longer hair requires the generation and rendering of longer line segment geometries, while the noise used to create curly hair adds disturbance to the geometric structure. Rendering curly hair requires a longer computation time than rendering longer hair. The combination of curly and long hair setups adds the most stress to the proposed system, thus achieving the lowest average frame rate and highest average frame time.

The last test is done with increasing number of guide hair strands of 400, 500, 600, 700 and 800 to increase the total number of hair strands to 25600, 32000, 38400, 44800 and 51200 hairs respectively, where number of guide hairs within each guide patch are kept constant at 64 strands. Results are as shown below.

TABLE VI. AVERAGE FRAME RATE AND FRAME TIME OF SIMULATION UNDER DIFFERENT TOTAL NUMBER OF STRANDS USED IN 60 SECONDS.

Total Number of Hair Strands	Frame Rate (fps)	Frame Time (ms)
25600	56.02	17.85
32000	47.17	21.20
38400	39.49	25.32
44800	34.64	28.87
51200	31.72	31.53

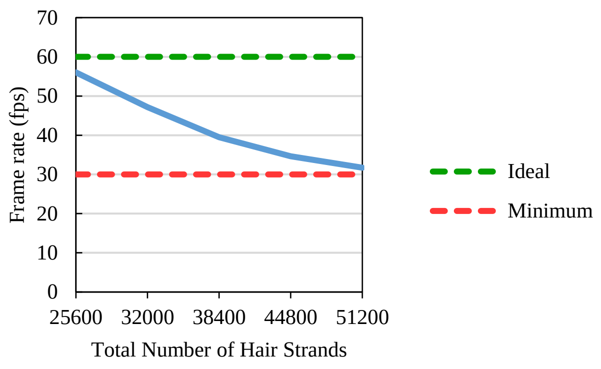

Fig. 8. Rendering performance under increasing total number of hair strands used.

From the results above, the frame rate decreases and frame time increases as more total strands of hair are rendered in the scene, where 25600 total strands achieved the highest performance of an average 56.02 fps and 17.85 ms while 51200 strands achieved the lowest performance of an average 31.72 fps and 31.53 ms. This is because more GPU passes are required as more hair strands are required to be rendered, thus dropping the performance of the shader. However, the proposed system still achieves real-time performance, even when rendering 51200 hair strands. This allows leeway for proposed system to account for hair transparency.

Therefore, the implementation of hair interpolation improves the hair rendering performance, where interpolation has allowed the proposed system to render hair strands in real-time with different hair type setups and with different total hair strands of up to about 50000 strands.

B. *Visual Result of Depth Peeling*

This section discusses the visual result of the inclusion of hair transparency within the proposed system and the differences between traditional alpha blending technique and depth peeling technique.

Fig. 9 to Fig. 13 show the visual results of rendered hairball model, with and without accounting for transparency properties. The visual appearance of hair without proper transparency looks dull and lacks depth, as illustrated in left figures in Fig. 12 and Fig. 13. In contrast, rendered models that incorporate transparency, like in the right figures in Fig. 12 and Fig. 13, produce significantly improved visual representation. However, Fig. 10 shows why traditional alpha blending is not suitable to be implemented when rendering hair strands. This is because hair strands are highly overlapping with one another as discussed in Section II. This property of hair leads to renders with incorrect occlusion as strands are only sorted solely based on the root's z-depth values, rendered, and then blended together, resulting in an average of all the contributing hair fragments.

Multi-pass rendering approach, depth peeling technique results in visually better hair renders as shown in Fig. 11. The final rendered image accounts for transparency along with the sorting of hair strands. This is because the algorithm sorts the scene per fragment along the view direction by peeling the hair geometry in layers and then blending them together. This technique allows a full sorting and blending of all strand fragments.

Depth peeling also acts as an anti-aliasing technique in the algorithm to overcome aliasing edges caused by sub-pixel-sized hair geometry. This can be shown as the right figures in Fig. 12 and Fig. 13 have smoother edges that look less aliased or jagged as compared to the left figures. Rough and aliased edges are reduced through the blending effect when layers produced from the geometry peeling are blended to form the final image.

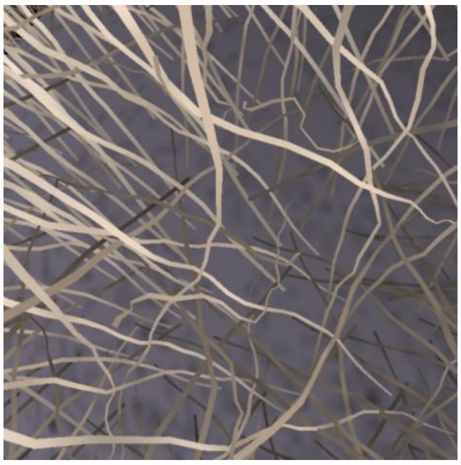

Fig. 9. Zoomed in frame of rendered hairball model using Marschner shader without translucent effect.

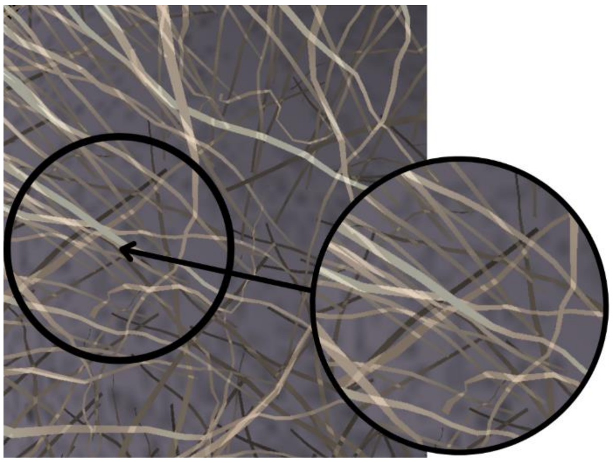

Fig. 10. Zoomed in frame of rendered hairball model using Marschner shader and alpha blending transparency. Hair geometry is rendered with incorrect occlusion as hair fragments are not sorted per pixel.

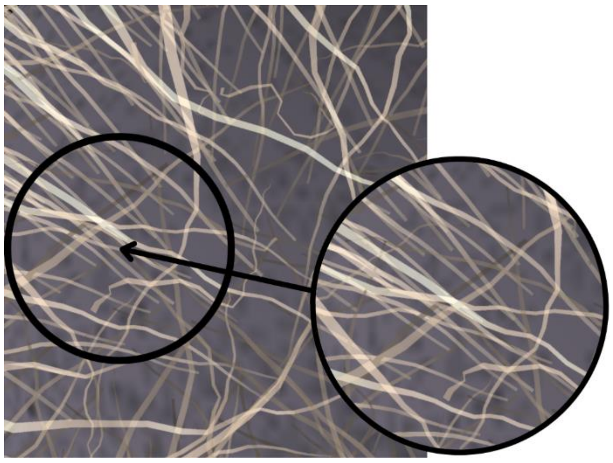

Fig. 11. Zoomed in frame of rendered hairball model using Marschner shader and depth-peeling transparency. Hair geometry is rendered after sorting hair fragments per pixel.

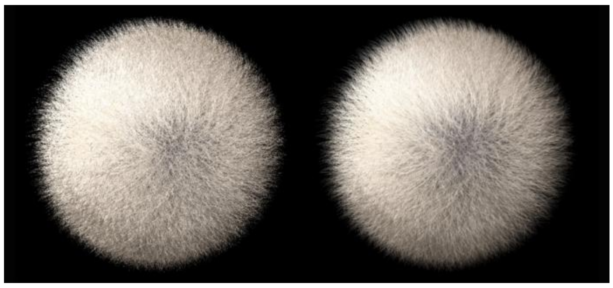

Fig. 12. Full Marschner hairball model rendered without (left) and with (right) depth peeling.

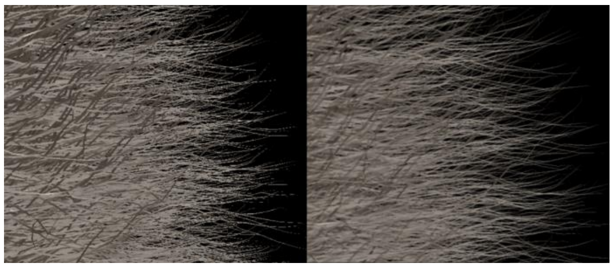

Fig. 13. Close-up view of Marschner hair rendered without (left) and with (right) depth peeling.

Therefore, depth peeling has improved the visual result of hair transparency by producing correct occlusion of overlapping strands by sorting the hair fragments per pixel.

VI. Conclusion

This project has provided a technique to improve hair rendering performance and render explicit hair strands with correct occlusion. Results from hair rendering performance tests and visual comparison tests on depth peeling have shown that the proposed system has achieved the objectives of this project. Implementation of hair interpolation when rendering hairball model achieved real-time performance, and depth peeling handled incorrect occlusion by sorting overlapping strand fragments per pixel from camera view.

Acknowledgment

I would like to express my gratitude to my supervisor, Rechard Lee, for providing guidance and assistance throughout the period of my dissertation. I would also like to express my gratitude to my family and friends who provided me support during my final year project. This study would not have been possible without their assistance.

References

[1] K. Ward, F. Bertails, T.-Y. Kim, S. R. Marschner, M.-P. Cani, and M. C. Lin, "A survey on hair modeling: Styling, simulation, and rendering," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 2, pp. 213–234, 2007.

[2] E. Sintorn and U. Assarsson, "Real-time approximate sorting for self shadowing and transparency in hair rendering," in *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, 2008.

[3] S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan, "Light scattering from human hair fibers," *ACM Trans. Graph., vol. 22,* no. 3, pp. 780–791, 2003.

[4] J. Ou, F. Xie, P. Krishnamachari, and F. Pellacini, "ISHair: Importance sampling for hair scattering," *Comput. Graph. Forum,* vol. 31, no. 4, pp. 1537–1545, 2012.

[5] J. Hunz, "Interactive physically-based hair rendering," 2016.

[6] X. Yu, J. C. Yang, J. Hensley, T. Harada, and J. Yu, "A framework for rendering complex scattering effects on hair," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games,* 2012.

[7] Z. Wang et al., "HVH: Learning a hybrid neural volumetric representation for dynamic hair performance capture," *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6143–6154.

[8] E. S. V. Jansson, M. G. Chajdas, J. Lacroix, and I. Ragnemalm, "Real-time hybrid hair rendering." *The Eurographics Association*, 2019.

[9] J. T. Kajiya and T. L. Kay, "Rendering fur with three dimensional textures," *Comput. Graph. (ACM)*, vol. 23, no. 3, pp. 271–280, 1989.

[10] J. Gray, , Dawber, R., and C.Gummer, (1997). "The world of hair: a scientific companion. Macmillan."

[11] C. Everitt, "Interactive order-independent transparency," *Gamedevs.org.*

[12] A. Leshonkov and V. Frolov, "Real-time rendering of small-scale volumetric structure on animated surfaces," *Ceur-ws.org*.

[13] L. Bavoil, and K. Myers. "Order independent transparency with dual depth peeling," *NVIDIA OpenGL SDK*, 1, 12. 2008

[14] L. Carpenter, "The A -buffer, an antialiased hidden surface method," in *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 1984.

[15] L. Bavoil, S. P. Callahan, A. Lefohn, J. L. D. Comba, and C. T. Silva, "Multi-fragment effects on the GPU using the k-buffer," in *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 2007.

[16] E. Enderton, E. Sintorn, P. Shirley, and D. Luebke, "Stochastic transparency," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, 2010.*

[17] O'Reilly, E. (2022). HairRendering: An OpenGL application demonstrating realistic physics and lighting for hair. GitHub. https://github.com/elor1/HairRendering

[18] J. T. Chang, J. Jin, and Y. Yu, "A practical model for hair mutual interactions," in Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2002.