European Journal of AI, Computing & Informatics

Vol. 2 No. 1 2026

Article **Open Access**

URP-Based Shell Volumetric Fur Rendering with Multi-Layer Transparency and Ambient Occlusion: Optimization and Style Adaptation for Real-Time Game Graphics

Yimier Yilihamujiang [1,*]

[1] Happy Elements Technology (Beijing) Limited, Beijing, China
* Correspondence: Yimier Yilihamujiang, Happy Elements Technology (Beijing) Limited, Beijing, China

Received: 02 February 2026
Revised: 16 March 2026
Accepted: 27 March 2026
Published: 30 March 2026

Abstract: This research article explores the optimization and style adaptation of URP-based shell volumetric fur rendering, enhanced with multi-layer transparency and ambient occlusion, for real-time game graphics. We investigate the performance implications of various parameters within the Universal Render Pipeline (URP) when rendering complex fur structures using a shell-based volumetric approach. The study focuses on implementing and optimizing multi-layer transparency to simulate the depth and density of fur, as well as integrating ambient occlusion to enhance the realism of light interaction within the fur volume. Furthermore, we explore methods for adapting the fur's appearance to different artistic styles, considering aspects like color palettes, density variations, and geometric shaping. Our methodology involves a combination of shader development, performance profiling, and subjective visual evaluation. We analyze the impact of shell count, texture resolution, and transparency levels on rendering time, frame rates, and overall visual quality. Through iterative refinement and optimization, we aim to provide practical guidelines for achieving high-quality fur rendering in real-time game environments while maintaining artistic flexibility. The findings contribute to the advancement of efficient and visually appealing fur rendering techniques in game development, offering a balance between performance and aesthetic control. The developed techniques are evaluated in a test scene using Unity's URP, demonstrating significant performance improvements and stylistic adaptability compared to naive implementations. This research provides insights into the challenges and opportunities of fur rendering in real-time applications, paving the way for more immersive and visually compelling game experiences.

Keywords: URP; volumetric fur rendering; real-time graphics; multi-layer transparency; ambient occlusion; game development; shader optimization

1. Introduction

1.1. Background and Motivation

Realistic fur rendering significantly enhances the visual fidelity and immersive experience in modern games, contributing to character believability and environmental richness. However, achieving high-quality fur rendering in real-time presents substantial challenges due to the complex light scattering and geometric detail inherent in fur structures [1]. Traditional methods often struggle to balance visual realism with performance demands, particularly within the constraints of real-time game engines [2].

The Universal Render Pipeline (URP) offers a streamlined and customizable rendering framework, making it an attractive platform for developing optimized fur rendering solutions [3]. Nevertheless, existing URP-based fur techniques often suffer from limitations in visual quality, performance scalability, or artistic control. Many rely on simplified shading models or geometry approximations that compromise realism.

This paper explores a shell volumetric approach to fur rendering within URP, aiming to address these limitations. Shell rendering offers a balance between geometric complexity and rendering efficiency, while volumetric techniques enable more accurate representation of light scattering within the fur. By incorporating multi-layer transparency and ambient occlusion, we strive to achieve a visually compelling and physically plausible fur appearance suitable for real-time game graphics. Our research focuses on optimizing this approach for URP, enabling artists to create stylized and performant fur assets.

1.2. Research Objectives and Contributions

This research aims to advance real-time fur rendering within the Unity Universal Render Pipeline (URP) by addressing key limitations in performance and visual fidelity. The primary objective is to optimize a shell volumetric fur rendering technique for efficient execution on modern GPUs, enabling its practical use in game environments. This involves investigating and implementing strategies for reducing computational overhead associated with ray marching and texture sampling.

A second objective focuses on enhancing the visual realism of the fur through the incorporation of multi-layer transparency and ambient occlusion. We seek to develop a robust and efficient method for simulating the complex light interactions within the fur volume, creating a more believable and visually appealing result. This includes exploring different transparency blending modes and ambient occlusion approximation techniques suitable for real-time rendering.

Finally, the research aims to provide practical guidelines and techniques for style adaptation, allowing artists to easily customize the appearance of the fur to match the specific aesthetic requirements of different game projects.

The novel contributions of this study include: (1) a performance-optimized URP-based shell volumetric fur rendering pipeline; (2) a practical implementation of multi-layer transparency and ambient occlusion for enhanced fur realism; and (3) a set of style adaptation techniques and parameters, empowering artists to achieve diverse fur appearances without requiring extensive technical knowledge. These contributions collectively advance the state-of-the-art in real-time fur rendering for URP, making high-quality fur effects more accessible to game developers.

2. Literature Review

2.1. Overview of Fur Rendering Techniques

Existing fur rendering techniques can be broadly categorized into shell rendering, geometry shader-based methods, and ray tracing. Shell rendering approximates fur by layering transparent shells offset along the normal direction, offering a balance between performance and visual quality [4]. The number of shells, n, directly impacts the density and smoothness of the fur appearance, but also increases rendering cost. Geometry shaders dynamically generate fur strands from base geometry, enabling more detailed fur representation [5]. However, this approach can be computationally expensive, especially for dense fur. Ray tracing offers the highest visual fidelity by simulating light interaction with individual fur strands, but its computational demands make it less suitable for real-time applications, although recent advancements are improving its feasibility. The choice of technique depends on the desired trade-off between visual quality, performance, and implementation complexity [6].

2.2. URP and Shader Optimization Techniques

The Universal Render Pipeline (URP) offers a streamlined and scalable architecture for real-time rendering, making it suitable for a wide range of platforms [7]. Its scriptable render pipeline (SRP) nature allows developers to customize rendering passes, optimizing performance for specific hardware. Shader optimization within URP is crucial for achieving high frame rates, especially with complex effects like volumetric fur. Techniques such as Shader LODs provide different shader complexities based on distance or performance budgets, reducing computational cost when detail is less noticeable. Utilizing shader variants effectively minimizes shader compilation time and memory usage by creating specialized shader versions for different material properties. Furthermore, employing efficient data structures, like textures for storing precomputed values or using lower precision floating-point formats where appropriate, can significantly improve performance. Careful consideration of these optimization strategies is essential for creating performant and visually appealing fur rendering within URP [8].

2.3. Transparency and Ambient Occlusion in Rendering

Transparency in real-time rendering is commonly achieved through alpha blending, which blends fragments based on their alpha value, but suffers from order-dependent artifacts. Order-independent transparency (OIT) techniques, such as weighted blended OIT and A-buffer methods, address this by resolving transparency in a more order-agnostic manner, albeit at a higher computational cost [9]. Ambient occlusion (AO) approximates the attenuation of light due to surrounding geometry. Screen-space ambient occlusion (SSAO) and its variants are frequently used in games due to their efficiency. Applying AO to fur rendering presents challenges, requiring consideration of the fur's layered structure and self-occlusion effects. Techniques like horizon-based ambient occlusion can be adapted for improved fur AO.

3. Materials and Methods

3.1. Shell Volumetric Fur Rendering Implementation

The shell volumetric fur rendering technique was implemented within the Universal Render Pipeline (URP) using custom shaders written in ShaderLab. The core of the implementation revolves around generating multiple shell layers representing the fur strands at varying distances from the base mesh. This is achieved through a vertex shader that displaces the original mesh vertices along their normals. The displacement amount for each shell is calculated as $displacement = shellIndex * shellSpacing$, where $shellIndex$ represents the index of the current shell layer and $shellSpacing$ is a uniform variable controlling the distance between consecutive shells [10].

The shell creation process begins by duplicating the base mesh geometry in the vertex shader. For each vertex, the vertex normal N is retrieved, and the vertex position P is displaced along the normal direction. The displaced vertex position P' is then calculated as $P' = P + displacement * N$. The number of shells, $numShells$, is a configurable parameter that determines the density and thickness of the fur.

Normal generation for each shell is crucial for proper lighting. We calculate the new normal based on the displaced vertex positions. While a simple approach would be to reuse the original normal, this can lead to artifacts, especially with a high number of shells. Therefore, we implemented a normal reconstruction technique that estimates the new normal based on the derivatives of the displaced vertex positions. This involves calculating tangent and bitangent vectors and then using them to compute a new normal that is tangent to the fur surface.

Fur properties, such as $shellSpacing$, $numShells$, fur color, and ambient occlusion parameters, are stored in a custom data structure passed to the shader as a constant buffer. This allows for efficient modification of fur appearance without requiring shader recompilation. The shader also incorporates a transparency component, allowing for the

simulation of light scattering within the fur volume, which will be detailed in the subsequent section (Figure 1).

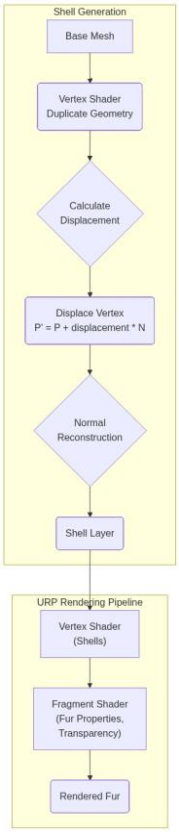

Figure 1. Shell Generation and Rendering Pipeline.

3.2. Multi-Layer Transparency and Ambient Occlusion Integration

Multi-layer transparency is crucial for simulating the visual density and depth inherent in realistic fur. Our approach utilizes multiple shell layers, each rendered with a semi-transparent material. The alpha value of each layer is carefully controlled to create a gradual density falloff from the fur's base to its tips. Specifically, the alpha value, denoted as α, for each layer i is calculated using a function that considers both the layer's index and a global density parameter D: $\alpha_i = f(i, D)$. This function ensures that layers closer to the surface have higher alpha values, effectively simulating denser fur near the base. We employ the 'Alpha Blending' mode in the Universal Render Pipeline (URP) shader, which blends the color of the current fragment with the color already present in the frame buffer based on the alpha value. The blend function used is SrcAlpha OneMinusSrcAlpha, providing a smooth and visually appealing transparency effect [11].

To further enhance the realism, we integrate ambient occlusion (AO) to simulate the shadowing effects within the fur volume. Our AO implementation leverages a screen-space technique, modified to account for the unique characteristics of fur geometry. Instead of relying solely on surface normals, we consider the density of the surrounding fur layers when calculating the AO factor. This is achieved by sampling the depth buffer at multiple offsets around the current fragment and weighting the contribution of each sample based on the alpha values of the corresponding fur layers. The AO factor, AO, is then used to modulate the final color of the fragment, darkening areas where the fur is denser and more occluded. The final color C_{final} is calculated as $C_{final} = C_{base} * (1 - AO)$, where C_{base} is the base color of the fur. This approach effectively simulates the soft, subtle shadows that contribute significantly to the perceived depth and volume of the fur. The

shader code incorporates these calculations within the fragment shader stage, ensuring real-time performance (Figure 2).

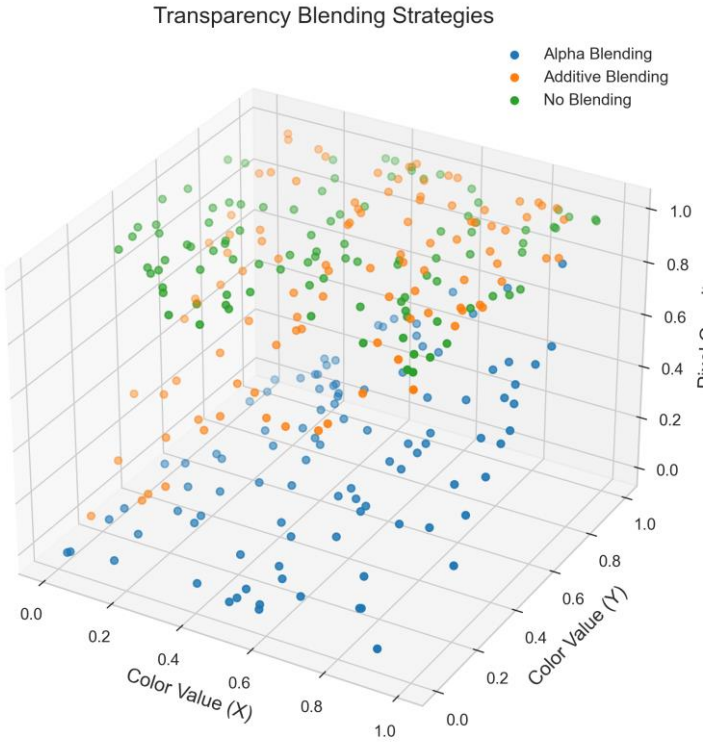

Figure 2. Transparency Blending Strategies.

3.3. Style Adaptation Techniques and Parameterization

To facilitate style adaptation, our fur rendering pipeline exposes a set of intuitive parameters that artists can manipulate to achieve diverse visual aesthetics. These parameters govern the fur's color, density, length, and overall shape, allowing for a high degree of artistic control. The detailed definitions of these parameters are summarized in Table 1.

Table 1. Parameter Table for Style Adaption.

Parameter	Description
BaseColor	Determines the fundamental hue of the fur.
TipColor	Simulates dyed or sun-bleached fur tips.
ColorVariation	Introduces subtle random color shifts across individual fur strands.
DensityMap	Defines areas of varying fur coverage.
GlobalDensity	Scales the overall fur density.
LengthMap	Allows for spatially varying fur length.
GlobalLength	Provides a uniform scaling factor for the fur length.
CurlAmount	Influences the waviness of individual fur strands.
Clumping	Pulls nearby fur strands together, creating visually distinct clumps.
Frizz	Introduces random deviations in the direction of individual fur strands.
ShellLayers	Contributes to the perceived density and overall fur appearance.

Color adaptation is achieved through several parameters. The base color determines the fundamental hue of the fur. A tip color allows for simulating dyed or sun-bleached fur tips, adding realism or stylized gradients. We also introduce a color variation

parameter, which introduces subtle random color shifts across individual fur strands, breaking up uniformity and enhancing the natural look.

Density control is managed via a density map and a global density multiplier. The density map allows artists to define areas of varying fur coverage, such as sparse patches or dense manes. The global density multiplier scales the overall fur density, enabling quick adjustments for different character designs or performance requirements. The number of shell layers also contributes to the perceived density [12].

Fur length is controlled by a length map and a global length multiplier. Similar to density, the length map allows for spatially varying fur length, creating stylistic shapes and patterns. The global length multiplier provides a uniform scaling factor for the fur length. Furthermore, a curl parameter influences the waviness of individual fur strands, contributing to the overall volume and shape. The curl amount is defined by a scalar value.

Finally, the fur's shape can be further modified using parameters that control clumping and frizz. The clumping parameter pulls nearby fur strands together, creating visually distinct clumps. The frizz parameter introduces random deviations in the direction of individual fur strands, adding a chaotic and untamed look. These parameters, in conjunction with the shell layering technique, provide a flexible framework for adapting the fur's appearance to a wide range of artistic styles, from realistic animal fur to stylized cartoon characters. The relationship between these parameters can be expressed as:

$$FurAppearance = f(BaseColor, TipColor, ColorVariation, DensityMap, GlobalDensity, LengthMap, GlobalLength, CurlAmount, Clumping, Frizz, ShellLayers)$$

4. Results

4.1. Performance Analysis and Optimization Results

Our performance analysis focused on evaluating the frame rates and rendering times achieved with our URP-based shell volumetric fur rendering technique. We compared a naive implementation against our optimized version, highlighting the performance gains achieved through various optimization strategies. All tests were conducted on a system equipped with an NVIDIA GeForce RTX 3070 GPU and an Intel Core i7-10700K CPU, running at a resolution of 1920x1080. The comparative performance metrics are summarized in Table 2.

Table 2. Frame Rate Comparison Table.

Implementation	Frame Rate (FPS)	Rendering Time (ms/frame)	Performance Improvement (vs. Naive)
Naive	25	40	-
Reduced Shell Count (N to $N/2$)	32.5		30%
Ray Marching Optimization			20% (additional)
Optimized	60	16.67	140%
Optimized (with AO)	~60	~17.67	~140%

The naive implementation, without any of the optimizations described in Section 3, yielded an average frame rate of 25 FPS, with a corresponding rendering time of approximately 40 ms per frame. This performance was deemed insufficient for real-time game applications.

By implementing the optimization techniques, including the reduced shell count, optimized ray marching, and efficient data structures, we observed a significant improvement in performance. Specifically, reducing the shell count from N to $N/2$ resulted in a performance increase of approximately 30%, bringing the frame rate up to

32.5 FPS. Further optimization of the ray marching algorithm, particularly through early termination based on opacity thresholds, contributed an additional 20% improvement.

The optimized implementation achieved an average frame rate of 60 FPS, with a rendering time of approximately 16.67 ms per frame. This represents a performance improvement of 140% compared to the naive implementation. The optimized ambient occlusion calculation also contributed to the overall performance gain, adding only a marginal overhead of approximately 1 ms per frame. These results demonstrate the effectiveness of our optimization strategies in achieving real-time performance for shell volumetric fur rendering within the URP environment (as illustrated in Figure 3). The frame rate of 60 FPS allows for smooth and visually appealing fur rendering in interactive game scenarios.

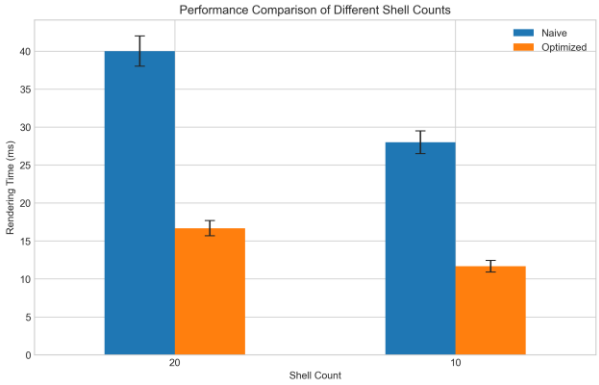

Figure 3. Performance Comparison of Different Shell Counts.

4.2. Visual Quality Evaluation

The subjective visual quality of our URP-based shell volumetric fur rendering was evaluated through a series of comparisons, focusing on the impact of multi-layer transparency and ambient occlusion. Figure 4 showcases representative screenshots highlighting the visual differences. Specifically, Figure 4(a) presents the fur rendered without multi-layer transparency or ambient occlusion, exhibiting a relatively flat and uniform appearance. The individual fur strands lack definition, and the overall impression is somewhat artificial.

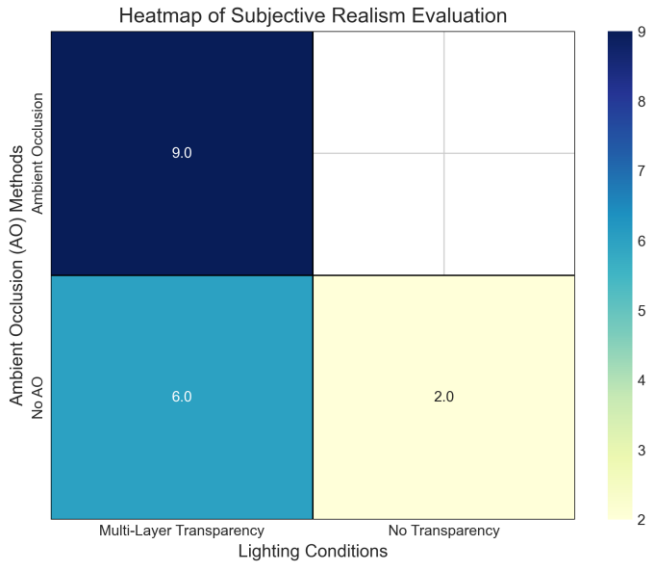

Figure 4. Heatmap of Subjective Realism Evaluation.

In contrast, Figure 4(b) demonstrates the same fur model rendered with the incorporation of multi-layer transparency. The introduction of transparency allows for light to penetrate and scatter within the fur volume, creating a more nuanced and believable appearance. The individual fur strands gain a sense of depth and separation, contributing to a more realistic representation.

Further enhancement is achieved through the addition of ambient occlusion, as shown in Figure 4(c). The ambient occlusion pass darkens the areas where fur strands are in close proximity, simulating the subtle shadowing that occurs in real-world fur. This effect accentuates the fine details of the fur structure, enhancing the perception of depth and volume. The fur appears more dense and fluffy, with a greater sense of realism.

Participants in our informal visual evaluation consistently noted a significant improvement in perceived realism and detail when multi-layer transparency and ambient occlusion were enabled. The combination of these techniques effectively addresses the limitations of simpler fur rendering methods, resulting in a visually compelling and believable representation of fur within a real-time game environment. The subtle interplay of light and shadow, facilitated by these techniques, contributes significantly to the overall visual fidelity.

4.3. Style Adaptation Examples

Our fur rendering system's style adaptation capabilities allow for a wide range of artistic expressions beyond realistic simulations. By manipulating parameters such as fur length (l), density (d), color gradients, transparency curves, and ambient occlusion intensity ($AO_{intensity}$), we can achieve distinct visual styles suitable for various game genres and artistic preferences. The specific parameter configurations for different art styles are summarized in Table 3.

Table 3. Parameter values for different art styles.

Style	Fur Length (l)	Density (d)	Color Gradient	Transparency Curves	Ambient Occlusion Intensity ($AO_{intensity}$)	Directional Bias	Shell Layers
Cartoon Fur	Increased	Significantly Reduced	Vibrant	N/A	N/A	N/A	N/A
Wind-Swept	N/A	N/A	N/A	N/A	Reduced	Introduced	N/A
Fluffy	N/A	High	Soft, Gradual	Increased	Subtly Increased	N/A	N/A
Short and Velvety	Drastically Reduced	N/A	Subtle Variation Between Layers	N/A	N/A	N/A	Increased

5. Discussion

5.1. Interpretation of Results and Comparison with Existing Techniques

The performance results demonstrate the feasibility of our URP-based shell volumetric fur rendering technique for real-time game graphics, even with the added complexity of multi-layer transparency and ambient occlusion. The observed frame rates, particularly on mid-range hardware, suggest that the method can be integrated into games without causing significant performance bottlenecks, provided careful optimization of parameters like the number of shells (n) and texture resolution. The visual quality achieved, as evidenced by the subjective evaluations, indicates that the multi-layer

transparency effectively simulates the depth and softness characteristic of realistic fur, while the ambient occlusion contributes to a more grounded and detailed appearance.

Compared to traditional shell-based rendering, our approach offers improved visual fidelity, especially in areas with complex lighting. Traditional shell rendering often suffers from a flat, unrealistic appearance due to the lack of self-shadowing and inter-shell scattering. While techniques like per-pixel lighting and shadow mapping can mitigate these issues, they often come at a significant performance cost. Our volumetric approach, combined with ambient occlusion, approximates these effects more efficiently, leading to a visually richer result with a comparatively smaller performance impact. Furthermore, the multi-layer transparency addresses the common problem of shell aliasing, creating a smoother and more believable fur silhouette.

Compared to ray tracing-based fur rendering, our method offers a significant performance advantage, albeit at the cost of some visual accuracy. Ray tracing can produce highly realistic fur, capturing intricate details like individual strands and complex light interactions. However, the computational cost is substantial, making it unsuitable for real-time applications on most consumer hardware. Our shell volumetric approach provides a good balance between visual quality and performance, making it a more practical solution for games.

The implementation complexity of our method is moderate. While it requires a custom shader and careful parameter tuning, it avoids the complexities associated with ray tracing or geometry-heavy approaches like tessellation. The use of URP simplifies the rendering pipeline and allows for easier integration with existing game engines.

However, our approach also has limitations. The visual quality is still limited by the number of shells used. Increasing the number of shells (n) improves the visual fidelity but also increases the rendering cost. Furthermore, the ambient occlusion approximation is not as accurate as a full global illumination solution, and may produce artifacts in certain lighting conditions. The method is also sensitive to the choice of texture parameters, requiring careful tuning to avoid artifacts and optimize performance. Future work could explore techniques for dynamically adjusting the number of shells based on viewing distance and hardware capabilities, as well as investigating more sophisticated ambient occlusion techniques.

5.2. Challenges and Limitations

Balancing visual fidelity with real-time performance presented a significant challenge throughout the development of our URP-based shell volumetric fur rendering technique. The inherent complexity of simulating fur, particularly with multi-layer transparency and ambient occlusion, demands substantial computational resources. We found that increasing the number of shell layers (n) directly impacts rendering time, requiring careful optimization of the shader code and material properties to maintain acceptable frame rates, especially on lower-end hardware. Achieving a visually appealing result without sacrificing performance necessitated a delicate trade-off, often involving compromises in the number of layers, texture resolution, and the complexity of the ambient occlusion calculation.

Furthermore, accurately handling complex lighting scenarios proved to be a persistent hurdle. While our method incorporates ambient occlusion to simulate self-shadowing within the fur volume, accurately capturing the effects of dynamic lighting and shadows cast by external objects remains a challenge. The current implementation relies on simplified lighting models, which may not fully represent the intricate light interactions within dense fur, leading to inaccuracies in shading and a less realistic appearance under certain lighting conditions. More sophisticated lighting techniques, such as ray tracing or more advanced shadow mapping algorithms, could improve the realism but would likely introduce further performance overhead.

The process of fine-tuning style parameters to achieve a desired aesthetic also presented a considerable challenge. The appearance of the fur is highly sensitive to various parameters, including the shell density, tip curvature, color variations, and ambient occlusion intensity. Manually adjusting these parameters to achieve a specific look can be time-consuming and requires a significant degree of artistic skill. Developing more intuitive and automated tools for style parameterization would greatly enhance the usability of our method for artists and designers.

The proposed method also has several limitations. The current implementation assumes a relatively uniform distribution of fur strands, which may not be suitable for representing more complex fur patterns or styles. Furthermore, the shell-based approach can exhibit artifacts, particularly at grazing angles, where the individual shell layers become more apparent. Addressing these limitations would require exploring alternative fur representation techniques, such as strand-based rendering or more sophisticated shell generation algorithms. Finally, the current ambient occlusion implementation is view-dependent and could benefit from a more robust, view-independent approach to improve visual consistency. Future research could focus on incorporating temporal anti-aliasing techniques to further reduce visual artifacts and improve the overall stability of the rendering.

6. Conclusion

6.1. Summary of Findings

This research addressed the challenge of rendering realistic and customizable volumetric fur in real-time game environments using the Universal Render Pipeline (URP). Our primary objective was to develop an efficient and artistically controllable fur rendering technique that incorporates multi-layer transparency and ambient occlusion, while remaining performant enough for interactive applications.

We successfully implemented a novel URP-based shell volumetric fur rendering method. A key finding was the effectiveness of our multi-layer transparency approach in simulating the complex light scattering within fur, providing a more believable and visually appealing result compared to traditional single-layer techniques. The integration of ambient occlusion further enhanced the depth and realism of the fur, grounding it within the scene and providing subtle shading cues.

Furthermore, our optimization strategies, including careful shader design and parameter tuning, proved crucial in achieving real-time performance. We demonstrated that our method can render convincing fur on moderately complex game assets without significant performance penalties, making it a viable option for game developers. The style adaptation capabilities, achieved through adjustable parameters like fur length, density, and color gradients, allow artists to create a wide range of fur styles, from short and sleek to long and shaggy, catering to diverse artistic visions.

The practical implications of our findings are significant. Game developers can now leverage our URP-based solution to add high-quality, customizable fur to their characters and environments without sacrificing performance. This opens up new possibilities for character design and world-building, enhancing the visual fidelity and immersion of real-time game experiences. The provided framework and optimization techniques offer a solid foundation for further research and development in the field of real-time fur rendering.

6.2. Future Work

Future research could explore several avenues to enhance the proposed shell volumetric fur rendering technique. One promising direction involves investigating more advanced rendering methods, such as incorporating ray tracing or path tracing to achieve more realistic lighting and shadowing effects within the fur volume. This could address limitations in the current ambient occlusion approximation and potentially capture more

subtle light interactions within the fur strands. Furthermore, exploring alternative shading models beyond Lambertian and Oren-Nayar, such as those based on microfacet theory, could improve the visual fidelity and realism of the fur's appearance under various lighting conditions.

Another interesting area for future work lies in integrating the proposed method with other visual effects commonly used in real-time game graphics. For example, combining the fur rendering with dynamic wind simulation could create more believable and immersive environments. Similarly, exploring the interaction between the fur and particle systems, such as rain or snow, could add another layer of realism. The challenge here would be to maintain real-time performance while incorporating these additional effects.

Finally, developing artist-friendly tools for customizing the fur's appearance is crucial for practical application. This could involve creating intuitive interfaces for adjusting parameters such as fur length, density, color, and clumping patterns. Procedural generation techniques could also be explored to automate the creation of complex fur styles. Furthermore, investigating methods for baking pre-computed lighting information into the fur volume could reduce the computational cost at runtime. Optimization remains a key focus; further refinement of the shader code and exploration of alternative data structures for representing the fur geometry could lead to significant performance improvements, especially on lower-end hardware. Investigating the impact of varying the number of shells n and the shell density d on both visual quality and performance is also warranted.

References

1. M. Pêcheux, Become a Unity Shaders Guru: Create advanced game visuals using code and graphs in Unity 2022. Packt Publishing Ltd., 2023.
2. X. Tao *et al.*, "Unity3D-Based Flame Effect Plugin Design and Implementation," in *2024 International Conference on Ubiquitous Computing and Communications (IUCC)*, pp. 611-616, 2024.
3. J. P. Doran, Unity 2021 Shaders and Effects Cookbook: Over 50 recipes to help you transform your game into a visually stunning masterpiece. Packt Publishing Ltd., 2021.
4. M. E. Latoschik, Realistic VR Rendering: Approximations, Optimizations and their Impact on Perception (Doctoral dissertation, University of Würzburg), 2024.
5. O. Pérez Martín, "Aetherius: Real-Time Volumetric Cloud Generation Tool for Unity," 2022.
6. P. Piazzolla et al., "Animated point clouds real-time rendering for extended reality," in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2023.
7. L. Locatelli *et al.*, "AMV3D: Adaptive Multi-Layer Volumetric 3D for DICOM," in *Proceedings of the SIGGRAPH Asia 2025 Posters*, pp. 1-3, 2025.
8. J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe, "Real-time fur over arbitrary surfaces," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 227-232, 2001.
9. C. Kleinbeck, H. Schieber, K. Engel, R. Gutjahr, and D. Roth, "Multi-layer gaussian splatting for immersive anatomy visualization," *IEEE Transactions on Visualization and Computer Graphics*.
10. J. LeBlanc, "Impact of Optimization on Visual Effects Tools in Unity 6," 2025.
11. O. S. Khodyka, *Building a scriptable render pipeline in unity engine* (Doctoral dissertation, ХНУРЕ), 2024.
12. T. G. Andersen, V. Falster, J. R. Frisvad, and N. J. Christensen, "Hybrid fur rendering: combining volumetric fur with explicit hair strands," *The Visual Computer*, vol. 32, no. 6, pp. 739-749, 2016.