Final Research Project Report (Computer Graphics, COMP8610-2024)
Enhanced Real-Time Grass Rendering with Advanced Tessellation and Dynamic Wind Simulations

Han Yan	Wenxiao Wu	Tianyu Wang
u7618352	u7543073	u7528058

The Australian National University

Abstract

Grass rendering is essential for creating realistic outdoor scenes in real-time applications, yet rendering each blade as geometry has traditionally been too computationally expensive.Traditional image-based techniques often lead to visual artifacts and fail to capture the dynamic interactions of grass with environmental factors. This paper presents an advanced method for real-time grass rendering using tessellation shaders and compute shaders. Each blade of grass is represented as a tessellated quad, allowing for detailed geometric representation and dynamic interaction with wind and objects. By utilizing specialized textures for encoding grass density, height, and appearance, along with efficient culling methods, our approach achieves high visual fidelity and performance.

1 Introduction

Rendering realistic grass is essential for enhancing the visual fidelity of outdoor scenes in virtual reality and computer games. Traditional methods, such as textures mapped onto the ground or transparent billboards, often produce unsatisfactory results, especially when viewed from various angles. This paper introduces a technique leveraging tessellation shaders to render grass blades as actual geometry, providing higher realism and enabling dynamic environmental interactions. Each grass blade is dynamically tessellated and animated based on environmental interactions, providing a higher degree of realism.

2 Motivation and Problem Statement

Real-time rendering of realistic grass is a critical aspect of creating immersive virtual environments, particularly in applications like video games, simulations, and virtual reality. The intricate details of grass, with its countless blades, each reacting to environmental factors such as wind, gravity, and collisions, add significantly to the visual realism of outdoor scenes. However, achieving this level of detail in real time has been a significant challenge due to the high geometric complexity involved. Traditional rendering methods, which often rely on texture mapping or billboards, simplify the rendering process but at the cost of realism, resulting in noticeable artifacts and a lack of dynamic interaction.

This project aims to overcome these limitations by introducing a method that renders grass as tessellated geometry, allowing for detailed and dynamic grass fields that interact with environmental factors such as wind and objects.

3 Previous Works

Several approaches have been developed to address the challenges of grass rendering, ranging from texture-based methods to more advanced geometric techniques:

Texture-Based Methods: Early methods, such as those by Shah et al.[1], used flat textures mapped onto the ground to represent grass. While these methods were computationally efficient, they resulted in poor visual quality when viewed from low angles. They improved this approach by introducing a bidirectional texture function and displacement mapping to enhance realism, though this came at the cost of increased rendering time.

Billboard Techniques: Techniques by Orthmann et al. utilized billboards to represent grass bunches, allowing for some environmental interactions[2]. However, these methods often resulted in flat appearances when viewed from above. Habel et al[3]. employed half-transparent texture slices arranged in a regular grid to improve visual quality, but visual artifacts still persisted under certain viewing conditions.

3D Texture and Geometry Approaches: Neyret extended fur rendering techniques to grass by using 3D textures[4], which provided high-detail geometry but often appeared artificial.

Instanced Rendering: Boulanger et al. combined various techniques, using instanced rendering and 3D textures to achieve different levels of detail[5]. Wang et al. developed a method that rendered grass fields by drawing multiple instances of a single blade with a skeleton for animation[6]. These methods improved performance and visual quality but still faced limitations in handling dynamic interactions and achieving high density.

Tessellation-Based Method: Jahrmann and Wimmer proposed a technique that renders dense grass fields entirely as smoothly shaped geometry using tessellation shaders[7]. This approach not only enhances visual quality but also allows each blade of grass to be individually influenced by environmental forces, offering a more realistic and dynamic rendering solution.

Our project combines techniques from "Interactive Grass Rendering Using Real-Time Tessellation" and "Responsive Real-Time Grass Rendering for General 3D Scenes," incorporating new ideas and methods to enhance real-time grass rendering further. This includes the use of accurate culling methods, adaptable rendering pipelines, and physical models evaluated for each blade of grass, enabling interactions with environmental factors such as gravity, wind, and collisions. This comprehensive approach provides a significant advancement in real-time grass rendering by balancing high visual fidelity with computational efficiency.

4 Our Method

Our project combines the techniques from "Interactive Grass Rendering Using Real-Time Tessellation" and "Responsive Real-Time Grass Rendering for General 3D Scenes," enhancing them with new ideas to create a comprehensive grass-rendering solution. By leveraging tessellation shaders and compute shaders, we enable each blade of grass to be rendered as a detailed geometric object, ensuring high visual fidelity and enabling dynamic interactions with environmental factors such as wind, gravity, and collisions. Key improvements include accurate culling methods, an adaptable rendering pipeline, and advanced wind simulations. The integration of these techniques allows us to render grass fields on arbitrary 3D models, not limited to height maps, providing greater flexibility and realism.

4.1 Shaders

Our grass-rendering algorithm involves several key shader components. The vertex shader (grass.vert) processes the initial vertices of the grass blades, applying transformations and passing necessary data to the tessellation control shader. This sets up the initial positions and attributes of the grass blades, preparing them for further processing. The tessellation control shader (grass.tesc) determines

the tessellation levels for each grass patch, controlling the density of tessellated vertices based on the distance to the camera and other factors. This shader prepares the data for the tessellation evaluation stage, ensuring that the level of detail is dynamically adjusted to maintain performance while enhancing visual fidelity.

The tessellation evaluation shader (grass.tese) computes the final positions of the tessellated vertices, shaping each grass blade according to control points and applying deformations due to environmental forces such as wind. This shader ensures smooth and realistic geometry for each blade of grass, providing high visual quality. Finally, the fragment shader (grass.frag) handles the final coloring of each pixel, including texture sampling, lighting, and environmental effects. It ensures that the visual appearance of the grass blades is realistic and dynamically responsive, adjusting colors and shading based on the input textures and environmental data.

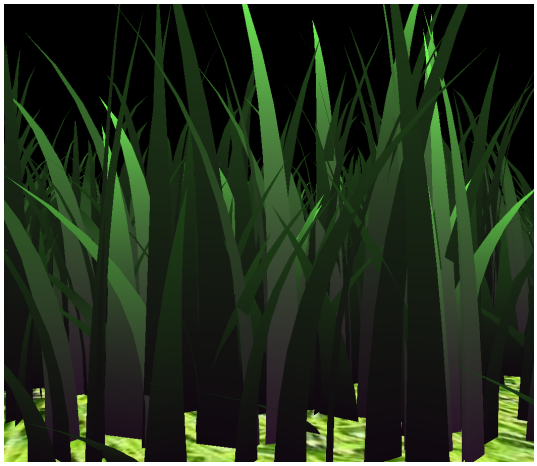

Figure 1: Grass Blade

4.2 Rendering

Our method includes some wind simulations to enhance the realism of grass movement. The compute shader is responsible for simulating physical forces on the grass blades, including three types of wind:

Strong Natural Wind: Simulates a consistent wind force blowing in a specific direction across the entire grass field. This wind provides uniform movement, simulating gentle breezes or strong gusts from a single direction.

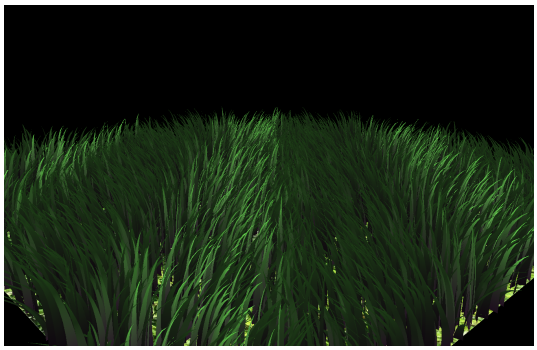

Figure 2: Strong Wind

Helicopter Wind: Simulates wind effects from a helicopter, with the wind direction dynamically changing based on the helicopter's position. The wind center moves in a circular pattern, adjusting for the changing wind direction and intensity near the helicopter.

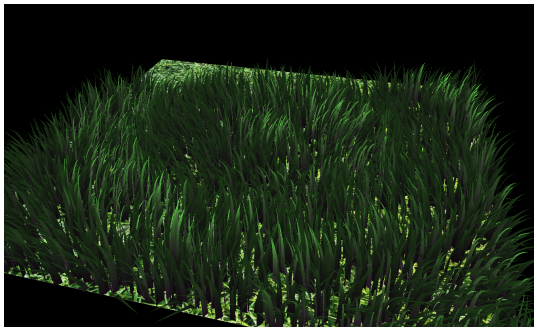

Figure 3: Helicopter Wind

Random Natural Wind: Adds variability and randomness to the wind force, creating a more natural and chaotic movement of grass blades. This simulates the unpredictable nature of wind in real environments, causing blades to sway and flutter.

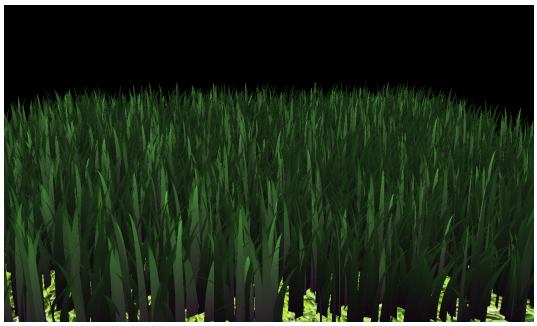

Figure 4: Natural random wind

Additionally, our method includes an accurate culling mechanism to render grass based on distance. Orientation culling checks if the blade is parallel to the view direction. View-frustum culling determines if the blade is within the view frustum. Distance culling checks if the blade exceeds the maximum visible distance. Grass patches closer to the camera are rendered with higher detail, while distant patches are rendered with lower detail to optimize performance without sacrificing visual quality.

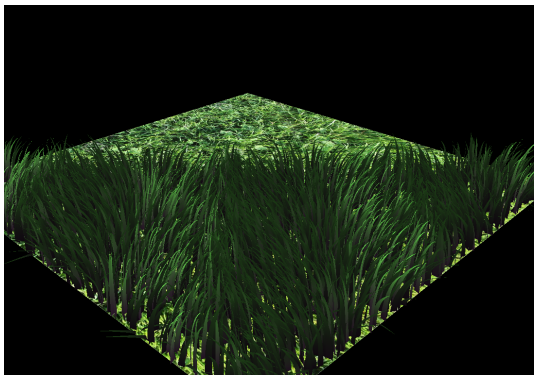

Figure 5: Render grass based on distance

5 Algorithms and Design

Vertex Shader (grass.vert) The vertex shader processes the initial vertices of the grass blades, applying transformations and passing necessary data to the tessellation control shader. This stage sets up the initial positions and attributes of the grass blades, preparing them for further processing in the tessellation shaders.

Tessellation Control Shader (grass.tesc) The tessellation control shader determines the tessellation levels for each grass patch, controlling the density of tessellated vertices based on the distance to the camera and other factors. This shader prepares the data for the tessellation evaluation stage, ensuring that the level of detail is dynamically adjusted to maintain performance while enhancing visual fidelity.

Tessellation Evaluation Shader (grass.tese) The tessellation evaluation shader computes the final positions of the tessellated vertices, shaping each grass blade according to control points and applying deformations due to environmental forces such as wind. This stage ensures smooth and realistic geometry for each blade of grass, providing high visual quality.

Fragment Shader (grass.frag) The fragment shader calculates the final color of each pixel, handling texture sampling, lighting, and environmental effects. It ensures that the visual appearance of the grass blades is realistic and dynamically responsive, adjusting colors and shading based on the input textures and environmental data.

Compute Shader (compute.comp) The compute shader is responsible for simulating physical forces on the grass blades, including advanced wind simulations with three types of wind: strong natural wind, helicopter wind, and random natural wind. It also simulates gravity and collisions. The compute shader updates the positions and states of the blades based on these interactions, ensuring realistic movement and response to environmental factors.

Gravity

The gravity applied to the blades is divided into two components:

1. Vertical component:
$$\vec{g}_E = 9.8 \times \text{normalize}(0, -1, 0)$$
2. Front directional component:
$$\vec{g}_F = 0.25 \times |\vec{g}_E| \times \text{normalize}(\vec{CurUp} \times \vec{widthdir})$$
3. Total gravitational force:
$$\vec{g} = \vec{g}_E + \vec{g}_F$$

Recovery Force

The recovery force, which helps the blade return to its original position, is calculated based on the blade's stiffness:
$$\vec{r} = (\vec{iv2} - \vec{CurV2}) \times \text{stiffness}$$
where:
$$\vec{iv2} = \vec{CurV0} + \text{normalize}(\vec{CurUp}) \times \text{bladeheight}$$

Wind Force

The wind force formulas vary based on different wind types. For deterministic wind:

1. Strong wind direction:
$$\vec{winddir} = \text{normalize}(1, 0, 1)$$
Helicopter wind direction:
$$\vec{winddir} = \text{normalize}(-1, 0, 1) + \vec{CurV0}$$

2. Wave coefficient:

$$\text{wavecoeff} = \cos\left(\frac{\vec{CurV0} \cdot \vec{wind\_dir} - \text{wind\_speed} \times \text{totalTime}}{\text{waveInterval}}\right)$$

3. Directional influence:

$$\text{fd} = 1 - |\vec{wind\_dir} \cdot \text{normalize}(\vec{CurV2} - \vec{CurV0})|$$

4. Height ratio:

$$\text{fr} = \frac{\vec{CurV2} - \vec{CurV0} \cdot \vec{CurUp}}{\text{bladeheight}}$$

5. Total wind force:

$$\vec{w} = \vec{wind\_dir} \times \text{wind\_power} \times \text{wavecoeff} \times \text{fd} \times \text{fr}$$

Random wind introduces variability:

1. Random natural wind direction:

$$\vec{wind} = \text{random3}(\vec{CurV0}) \times 3.0 \times \sin(\text{totalTime})$$

2. Directional influence:

$$\text{f\_d} = 1 - |\text{normalize}(\vec{wind}) \cdot \text{normalize}(\vec{CurV2} - \vec{CurV0})|$$

3. Height ratio:

$$\text{f\_r} = \frac{\vec{CurV2} - \vec{CurV0} \cdot \vec{CurUp}}{\text{bladeheight}}$$

4. Total wind force:

$$\vec{w} = \vec{wind} \times \text{f\_d} \times \text{f\_r}$$

Each wind model considers the blade's orientation relative to the wind direction (Directional Influence) and the blade's height (Height Ratio) to compute the overall impact of the wind. These factors ensure that the wind's effect is realistically modulated across different parts of the grass blade, contributing to natural-looking motion.

The direction of random natural wind is generated using a pseudo-random function (random3) based on the blade's initial position, modulated over time. This method introduces variability, making the wind's direction less predictable and more natural. Other two deterministic Wind use a more structured approach where the wind direction, speed, and effects are calculated based on fixed formulas, such as using cosine waves to simulate the wave effect of wind over time.

Total Force

The total force applied to the blade is given by:

$$\vec{tv2} = (\vec{g} + \vec{r} + \vec{w}) \times \text{deltaTime}$$

Final State

The new position of the blade's vertex V2 is calculated as:

$$\vec{fv2} = \vec{CurV2} + \vec{tv2}$$

This position is corrected with:

$$\vec{fv2} = \vec{fv2} - \vec{CurUp} \times \min(\vec{CurUp} \cdot (\vec{fv2} - \vec{CurV0}), 0)$$

The projection length l_proj is determined by:

$$\text{length}(\vec{fv2} - \vec{CurV0} - \vec{CurUp} \times (\vec{fv2} - \vec{CurV0}))$$

The new position of vertex V1 is:

$$\vec{fv1} = \vec{CurV0} + \text{bladeheight} \times \vec{CurUp} \times \max(1 - \frac{l\_proj}{\text{bladeheight}}, 0.05 \times \max(\frac{l\_proj}{\text{bladeheight}}, 1))$$

Calculating New Length

The initial length $L0$ is:

$$\text{distance}(\vec{fv}2, \vec{CurV}0)$$

The intermediate length $L1$ is:

$$\text{distance}(\vec{fv}2, \vec{fv}1) + \text{distance}(\vec{fv}1, \vec{CurV}0)$$

The final length L is given by [8]:

$$L = \frac{2 \times L0 + (n-1) \times L1}{n+1}$$

Updating Blade Position

Update middle vertex:

$$\vec{CurBlade}.v1.xyz = \vec{CurV}0 + \left(\frac{\text{bladeheight}}{L}\right) \times (\vec{fv}1 - \vec{CurV}0)$$

Update top vertex:

$$\vec{CurBlade}.v2.xyz = \vec{CurBlade}.v1.xyz + \left(\frac{\text{bladeheight}}{L}\right) \times (\vec{fv}2 - \vec{fv}1)$$

All models adjust their calculations based on dynamic properties such as the current time (totalTime) and the blade's position. This dynamic interaction is key to creating a lifelike simulation where the environment and the grass appear responsive to changing conditions.

These shaders collectively contribute to a robust grass rendering system that not only looks visually appealing but also interacts realistically with its environment, offering a dynamic and immersive experience in digital landscapes.

6 Discussions and Conclusion

Our project is building on and extending the foundational techniques introduced in "Interactive Grass Rendering Using Real-Time Tessellation" and "Responsive Real-Time Grass Rendering for General 3D Scenes." By integrating tessellation shaders and compute shaders, we achieve high visual fidelity and dynamic interactions with environmental factors, ensuring suitability for real-time applications. Key improvements in our method include advanced wind simulations, accurate culling methods, and an adaptable rendering pipeline, enabling detailed and realistic grass fields.

Compared to "Interactive Grass Rendering Using Real-Time Tessellation," our approach offers notable enhancements. While both methods utilize tessellation shaders for rendering grass blades and employ dynamic level of detail adjustments, our technique improves environmental realism through advanced wind simulations. We simulate three types of wind—strong natural wind, helicopter wind, and random natural wind—using compute shaders, resulting in more realistic and varied interactions. This ensures that grass blades respond dynamically to different wind patterns. Furthermore, our integration of compute shaders for physical simulations enhances the dynamic response and overall realism of the grass rendering.

In contrast with "Responsive Real-Time Grass Rendering for General 3D Scenes," our method introduces several advancements. Both approaches use accurate culling techniques to optimize performance by focusing on visually significant grass blades. However, our method further optimizes performance through an adaptable rendering pipeline that efficiently manages various shader stages and supports multiple levels of detail. This ensures optimal performance across different scenes and perspectives.

The integration of advanced wind simulations, detailed geometric rendering, and an adaptable rendering pipeline distinguishes our method from previous techniques, resulting in a more lifelike and dynamic grass rendering solution. Future work can explore further optimizations and enhancements, such as incorporating additional levels of detail and refining physical simulation models, to continue advancing the state of real-time grass rendering technology.

References

[1] Shah, Musawir A., Jaakko Kontinnen and Sumanta N. Pattanaik. “Real-time rendering of realistic-looking grass.” Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (2005): n. pag.

[2] Orthmann, Jens and Thomas Ertl. "GPU-based Responsive Grass."

[3] Habel, Ralf, Michael Wimmer, and Jirí Bittner. "Instant Animated Grass." Eurographics 2009.

[4] Neyret, Fabrice. “Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures.” IEEE Trans. Vis. Comput. Graph. 4 (1998): 55-70.

[5] Boulanger, Kevin et al. “Rendering Grass in Real Time with Dynamic Lighting.” IEEE Computer Graphics and Applications 29 (2009): n. pag.

[6] Wang, Chuan, Zhongqiang Wang, Qing Zhou, Chunxia Song, Yanyun Guan, and Qunsheng Peng. "Dynamic Modeling and Rendering of Grass Wagging in Wind: Natural Phenomena and Special Effects." (2005).

[7] Jahrmann, Klemens and Michael Wimmer. “Interactive Grass Rendering Using Real-Time Tessellation.” International Conference in Central Europe on Computer Graphics and Visualization (2013).

[8] Jahrmann Klemens , Wimmer Michael. (2017). Responsive real-time grass rendering for general 3D scenes. 1-10. 10.1145/3023368.3023380.

Confidential Peer Review

My name is Wenxiao Wu, a COMP8610 student for 2024S1. Our project (Enhanced Real-Time Grass Rendering with Advanced Tessellation and Dynamic Wind Simulations) has been completed by a 3-person team consisting of Han Yan, Wenxiao Wu and Tianyu Wang. From my best knowledge and assessment, I certify that the relative contributions among the team are:

Name and ID	Main duties and contributions	Contribution weighting
Han Yan: u7618352	Main coder, main video presentation designing and report writing.	34 %
Wenxiao Wu: u7543073	Second coder, optimise PowerPoint presentation and report reviewer.	33 %
Tianyu Wang: u7528058	Code reviewer, video recording and revise report.	33 %