Description

Animating foliage can be a complex task in game development and there is a number of different approaches to such a task. Some of them include giving an animation to each object, other use GPU instancing. To keep production time, visual appeal, and most importantly, performance at reasonable level, GPU instancing is widely used to to animate large quantities of foliage. This is the technique I chose to explore and test it's performance here.

Implementation

Project is crated in 2D, using the Godot game engine and making use of MultiMeshInstance2D. Foliage consists of only grass, each grass blade is a separate instance of a mesh. Wind is a noise texture scrolling through the world and displacing grass meshes accordingly. Each grass blade bends in shape of quadratic function, depending on wind strength at given time at that point. Color of the blades also shits slightly depending on wind, grass blades squash a bit as well. The wind texture is a variation of cellular noise with a few octaves and domain warping. Grass blades do not use a texture, instead have a slight gradient to them, also applied by the shader. Shader allows to adjust wind direction, speed, frequency, winds affect on vertical and horizontal aspect of grass movement and more.

Test setup

Tests were run all my personal laptop, IdeaPad 5i Pro Gen 6, equipped with Intel Core 11300H, NVIDIA GeForce MX450 and running a fairly clean install of Windows. The program was tested in a maximized window on a Full HD monitor, with no major programs running in the background. The frame rate has been unlocked to use 100% of graphics computing power.

For testing the performance I have written a custom profiler in Godot. Its job was to test every combination of 7 meshes and 7 amounts of grass blades. Each test was run for 15 seconds, then another mesh was substituted and that configuration was run for 15 seconds. After testing 7 meshes program increased the amount of grass blades to the next value and tested all the meshes again, 7 times. The whole process has been repeated 8 times in a row to minimize discrepancies between runs. To minimize the effects of thermal throttling on test results the program also run for 10 minutes before gathering data. The results of all eight runs of the profiler were averaged Unsaved to a .csv file. The entire profiling process took around two hours.

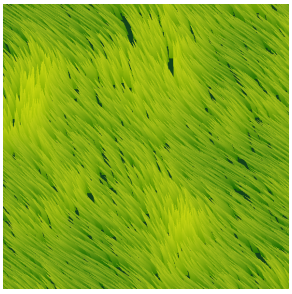
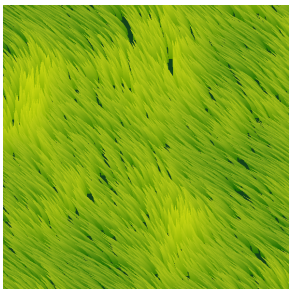
Placement of the grass meshes was excluded from tests but the placement was seeded to guarantee accurate results.

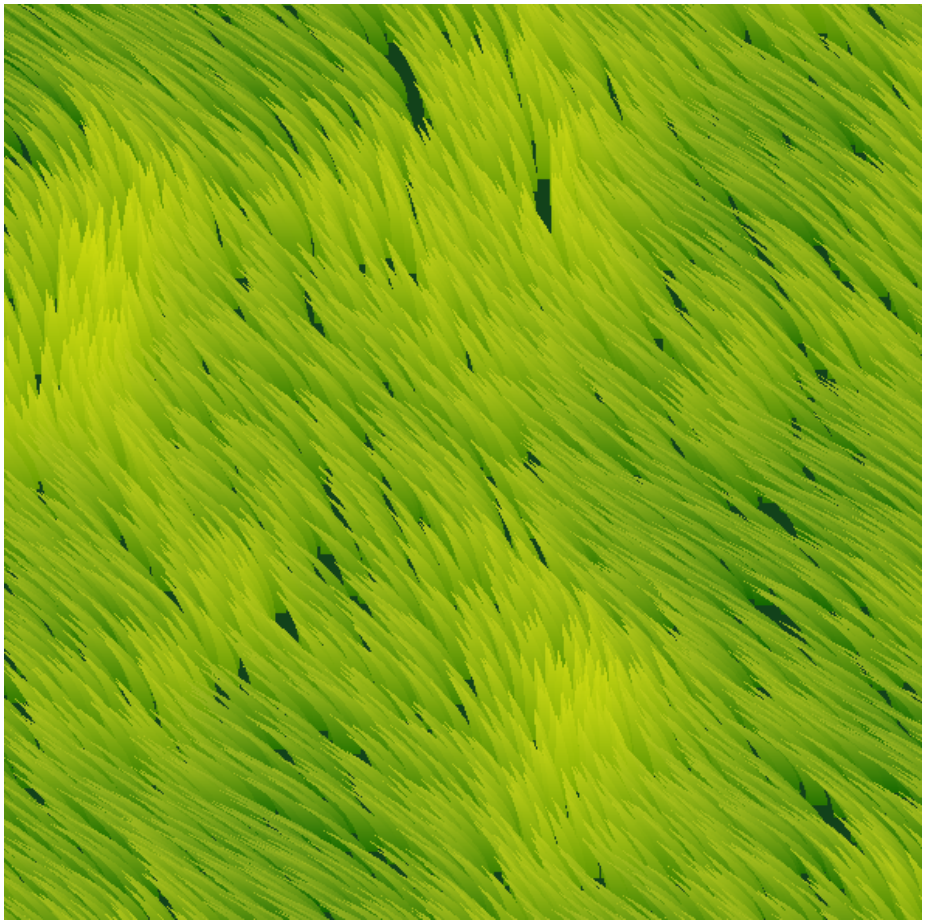
The base for the test was 17 vertices and 10,000 meshes; this is what I found

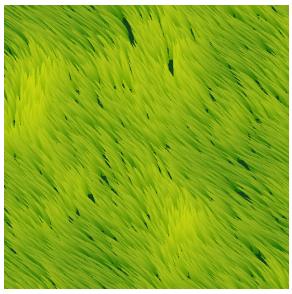
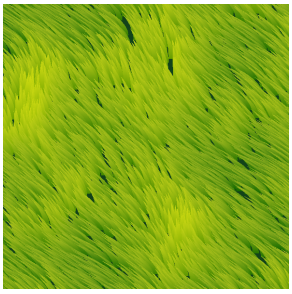
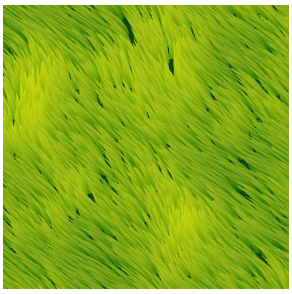
the most visually appealing. Other vertex counts and mesh counts were chosen to be 2, 4 and 8 times smaller and larger, making it seven each.

Comparing mesh complexity vs performance and visual appeal

I have created seven different meshes for the grass blades and tested each one for visual appeal and performance. Below are those seven different complexities of grass blade with corresponding vertex count.

3	5	9
		

17


33	65	129
		

As we can see the most drastic difference occurs between 3, 5 and 9 vertex meshes.

The 3 vertex mesh is too simple to create an illusion of realistic grass and the 5 vertex mesh still lacks complexity, potentially being even worse because of a jarring snap in the middle of each blade. On the 9 vertex mesh we can still notice snapping of the blade if we look for it, but it is smooth enough to create an illusion of smooth flowing grass. The 17 vertex mesh is the first one where individual edges cannot be discerned even if explicitly looked for. Beyond this point there is very little to none discernible difference between meshes, which makes measures that complex not useful in this context, though it can still be useful for other types of foliage.

Here we can see the results of tests performed on a 10,000 grass blade sample. The graph shows average frame time The pending on grass blade vertex count.

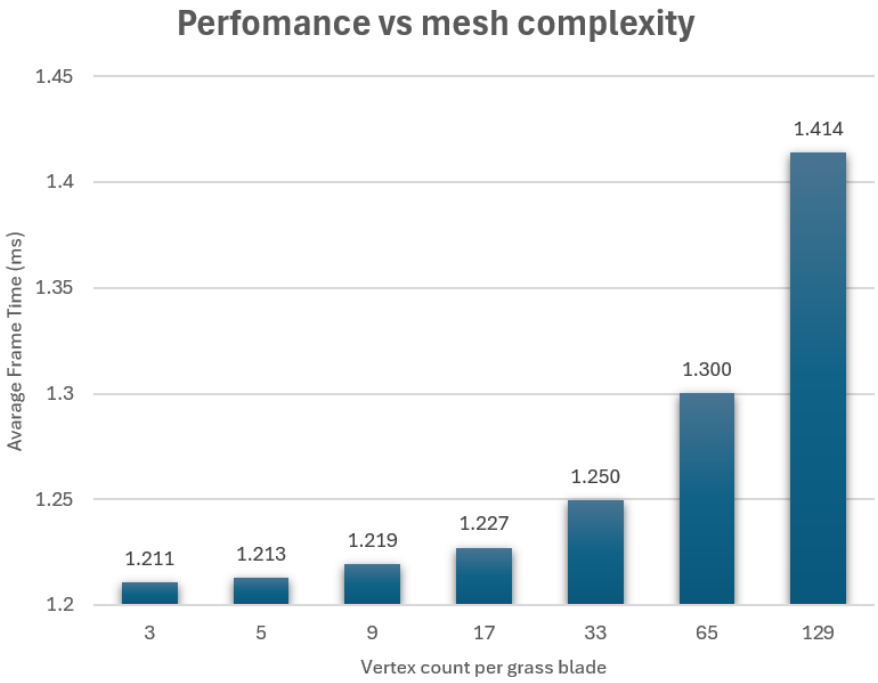


Figure 1: image

In line with our expectations we can observe higher computation times for more complex meshes. Computationally, there is virtually no difference between 3, 5 and 9 vertex meshes, while visual appeal suffers greatly from lower complexity meshes. On the higher end we can observe an inverse situation. While there is no visual difference between vertex count of 65 and 129 computationally this jump costs us around 0.7% of frame time budget for a 60FPS game, witch is not as insignificant as it may sound. The whole graph shows us that performance is more or less proportional to the amount of vertices. There is roughly twice

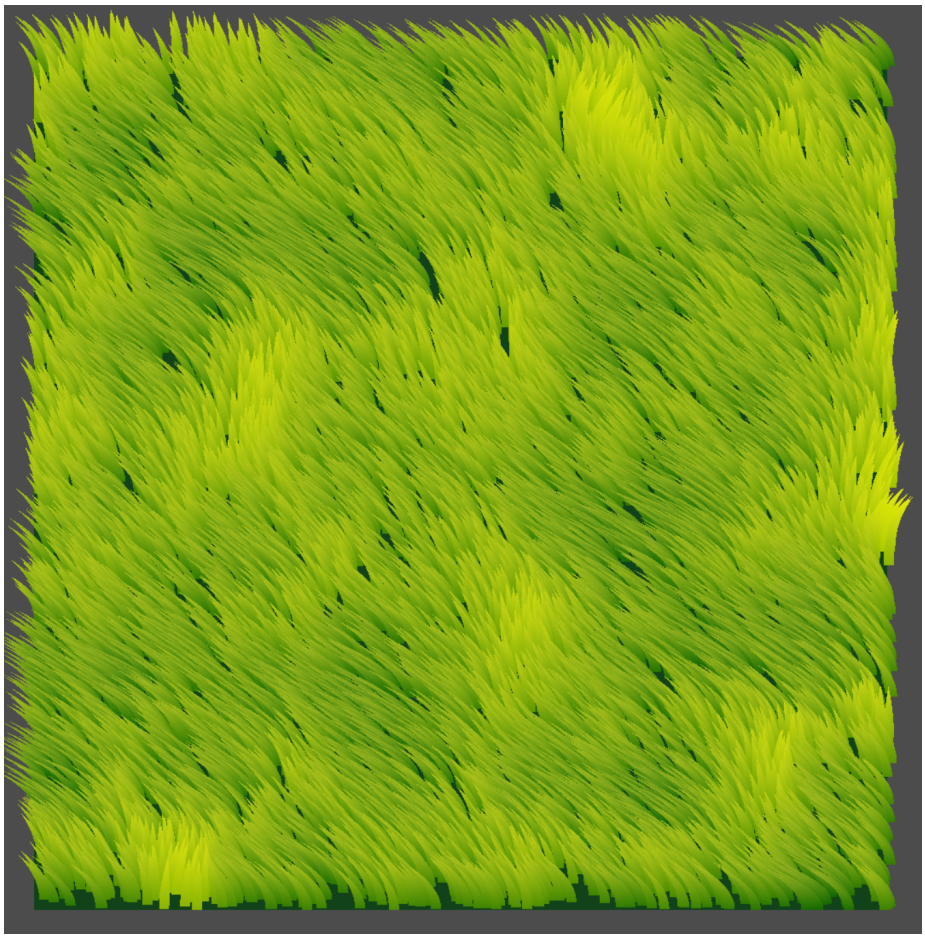
as much vertices in the 65 mesh than there is in 33 mesh and the jump in computation time is 50 s. Based on that, we should expect around a 100 s jump when going from 65 mesh to 129 mesh, that is indeed in line with the data.

Comparing grass blade count vs performance and visual appeal

Similarly to mesh complexity I have defined 7 values for amount of grass in the scene. Below are said 7 different amounts of grass with corresponding amounts given.

1,250	2,500	5,000
		

10,000



20,000 40,000 80,000

As we can see 1,250 and 2,500 grass blades are not enough to create a visually appealing image of a grass field with a mesh this small and simple. 5,000 grass blades for such an area seems to be a bare minimum here but still the image that it creates in this context is not convincing. 10,000 cross blades seem to create a good looking grass field where the ground is barely visible and the resulting image is not only satisfactory, but satisfying. At this density we can clearly distinguish between each grass blade and the effect of the wind looks natural. At 20,000 meshes ground beneath the grass is not visible anymore, grass is very dense and looks very smooth, almost hair-like. The image that it creates is also visually appealing. Although both 10,000 and 20,000 look very good, there is a clear difference in visual feel of the grass. Both amounts have different uses in different contexts. At 40,000 blades grass stops looking like grass and more closely resembles hair or fur. At 80,000 the problem only depends. Both amounts are unusable for foliage.

Below we can see the results of tests performed on above seen scenes filled with 17 vertex meshes of various amounts. The graph shows average frame time depending on mesh count.

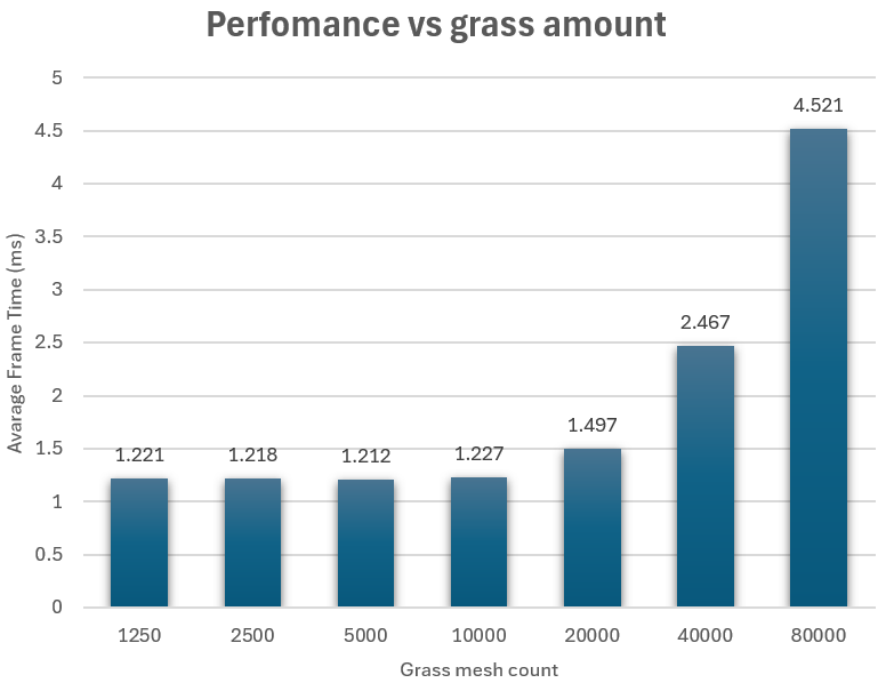


Figure 2: image

As we can see for the first 4 mesh counts difference in performance is negligible. Contradictory to what we expect within said amounts, 5,000 meshes seem

to Render quicker than 2,500 and 1,250, despite thorough testing methodology. This is most probably due to lack of precision and/or under the hood optimizations by the engine. Other causes could include fluctuations in Godot's other functionality aside from the shader, as well as lower densities of grass making each grass blade more visible therefore requiring more computation per grass blade, due to fragment shader. Whatever the case may be, it is clear that a 1000 more meshes does not impact processing time significantly. Based on that there is little purpose to lower mesh count below 10,000, as at that scale it creates a far inferior image and saves very insignificant amount of GPU time. Shifting our attention to the higher end of the scale the computational time is in line with what we would expect. The jump in computation time between 40,000 and 80,000 is around twice as large as between 20,000 and 40,000 indicating Linearly proportional correspondence between mesh count and average frame time. Aside from 40,000 and 80,000 giving an image not in line what we expect foliage to look like, it also takes an immense time to compute. Given that the time it takes for the engine to render one frame is no bigger than 1.3 milliseconds, 80,000 meshes taking additional 3 milliseconds to render is very expensive given the 16.6 millisecond frame budget.

Summary

Given everything that we have learned in this little study we can conclude that in case of this program we can comfortably set values to the amounts that look right and there is no need to sacrifice visual appeal, as the performance gains are negligible. Computational costs becomes an issue at a point that is above either our perception, in the case of mesh complexity, or visual esthetic we have been aiming for, in case of grass density.