Description

Shaders are a good resource that are used to define how materials and textures are rendered on the scene. Shaders allow developers to create specific and customized visual effects such as water, outlines for objects, dynamic illumination or in this case grass.

Grass shaders are a faster and more efficient solution in terms of performance than 3D models because it uses rendering techniques that allow the grass to be represented in a realistic and detailed way without having to load a large amount of data into the system's memory.

A 3D grass model is typically composed of a large number of polygons, each of which represents a small section of grass. When the model is rendered, the computer has to process each of these polygons to create the final image, which can be very costly in terms of computational resources. In addition, the 3D grass model cannot easily adapt to different types of terrain or to different lighting and shading conditions.

On the other hand, a shader uses advanced programming techniques to simulate the appearance of grass without the need to use a detailed 3D model. Instead, the shader uses algorithms to simulate the way the grass bends and sways in the wind, as well as the way light reflects off the blades of grass. These algorithms are highly optimized and can produce highly detailed and realistic images without requiring a large amount of computational resources.

With this in mind, there's an important aspect in order to make a more realistic shader, which is the interaction with the environment, so that when a character passes over the grass, the blades move around it. These calculations might affect performance depending on the number of characters walking through it simultaneously, so a series of tests are necessary to determine at what point the amount of characters have a significant impact on FPS. Besides that, there will also be four other variables that will be analyzed in order to determine its impact on FPS combined with the amount of characters on scene. The variables are Tessellation, Blade width, Blade height and Wind strength.

This shader has been made by using following tutorials:

- Unity Grass Shader Tutorial
- How To Make An Interactive Grass Shader

The code from the last tutorial has been adapted to interact with multiple characters at the same time. Also a simple code for the characters to walk around within the limits of the grass zone was implemented

Evaluation Proposal: Compare the impact of different variables and the amount of characters interacting with the shader on FPS.

Performing the tests

To compare all the shader variables in order to find a significant impact on FPS, they need to be compared to each other. All this data will be collected in an excel spreadsheet using the following tutorial:

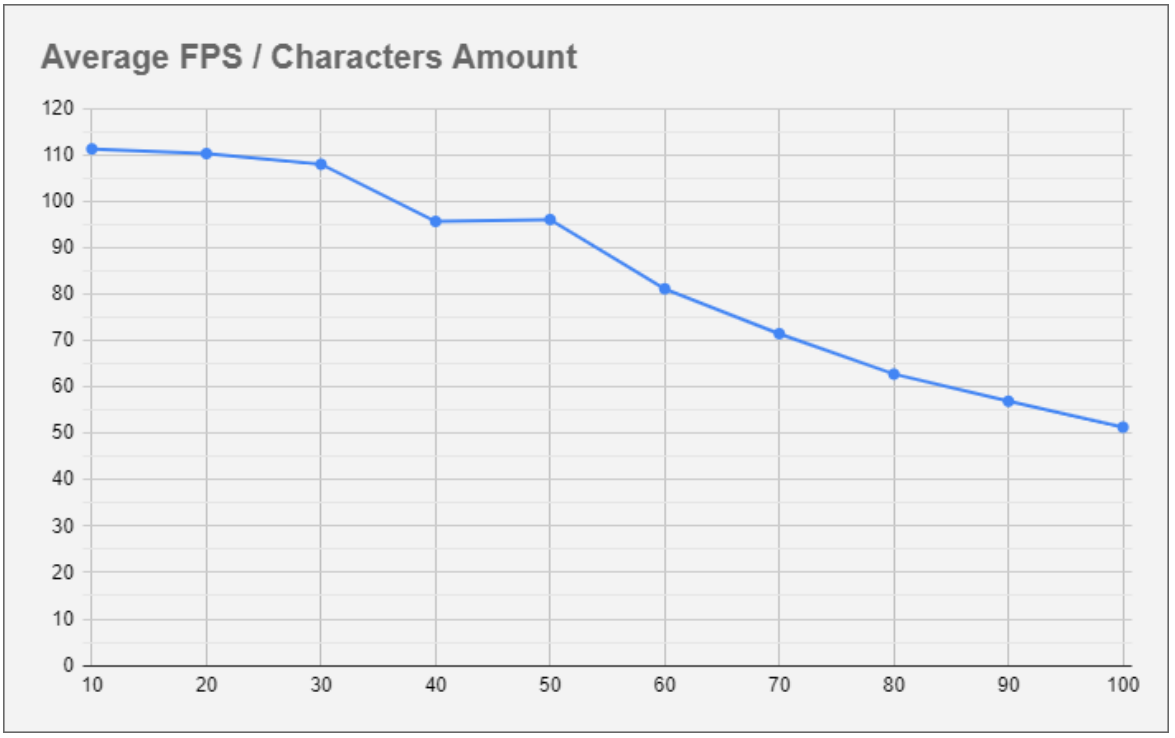
- Exporting data in Unity to open in Excel as a spreadsheet
- FPS Counter

The following setup was used to test:

- CPU: Intel(R) Core(TM) i7-9750H @ 2.60GHz
- GPU: NVIDIA GeForce RTX 2060
- RAM: 16,0 GB
- Motherboard: MS-17E5
- System Model: GL75 9SEK

Analysis

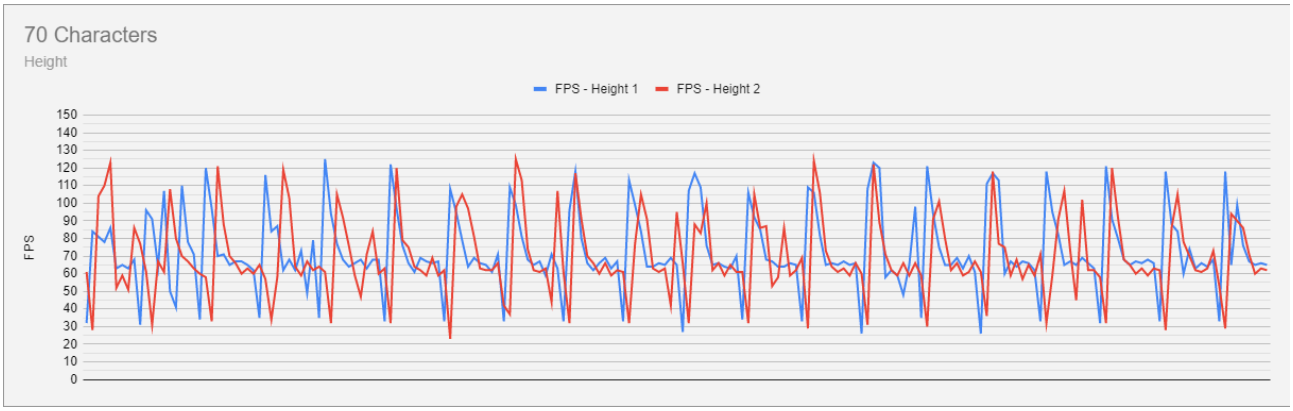
Initially, some "default" values were assigned to the grass shader in order to evaluate its performance only according to the number of characters in the scene. 100 characters were reached on the scene after 10 tests with 10-character increments each were conducted. The average FPS for each test was determined by recording a sample of 5 seconds for each test. With this initial test, the conclusion was that the limit of characters that the scene could handle before it had a significant impact on FPS were 70 to 80 characters, so these were the values on which the other shader parameters mentioned above were tested.



With the configuration for the 70 characters ready, several tests were made where each of the shader parameters were evaluated separately, in the case of the height, two tests were made with two different values: One with the height at 1 and the other with the height at 2, taking into account that the original value of this parameter was 0.5

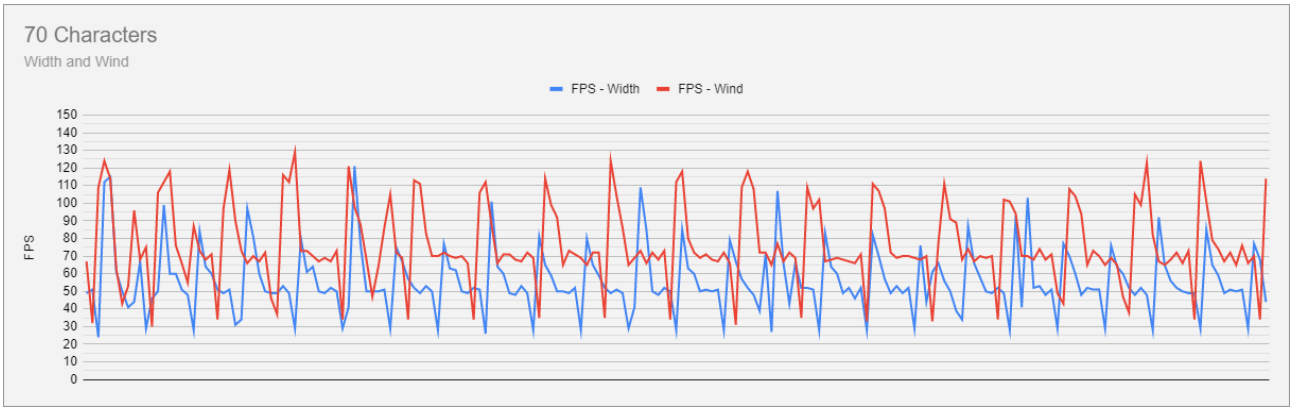
Neither test showed a significant impact on the FPS, and therefore both tests gave fairly similar results to the first test which was done with 70 characters and the default shader values.

70 Characters



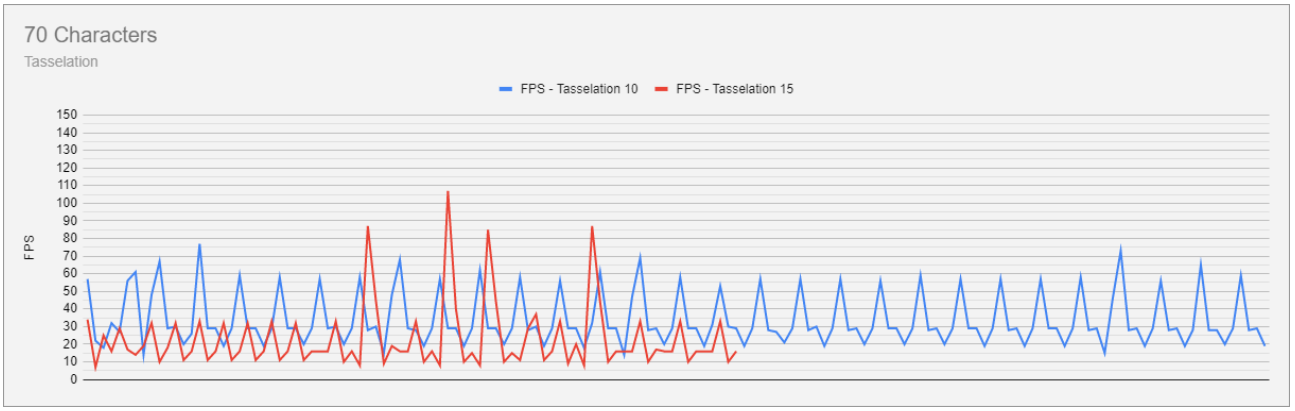
Average FPS for Heigth 1: 73 - Average FPS for Heigth 2: 69

After the tests in which both width of the grass blades and wind intensity were modified, the results were similar to those of the height modification and there was no significant change in the amount of FPS.



Average FPS for Width 1: 56 - Average FPS for Wind 0.7: 76

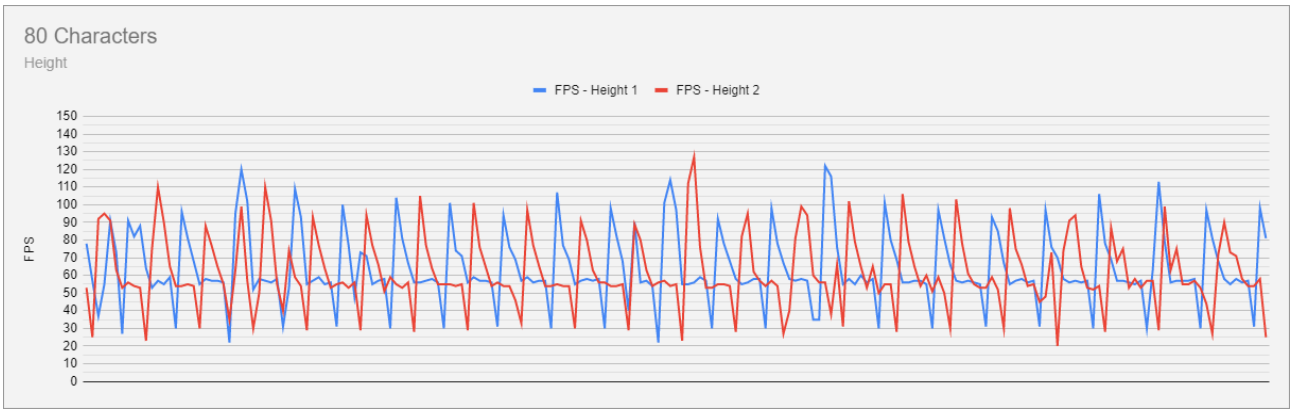
However, the tasselation level did have a highly significant impact on the FPS, as by splitting the mesh on which the shader was applied in order to add more grass blades, the rendering calculation along with the distortion caused by the characters walking arround is highly increased, causing the FPS to drop drastically and making it not recommended to apply to a real project with the values used for the test.



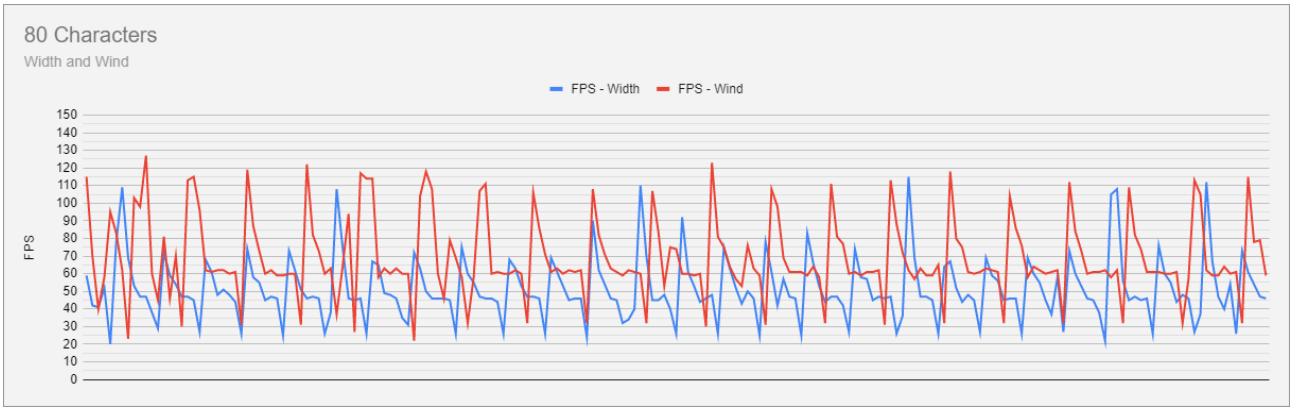
Average FPS for Tasselation 10: 34 - Average FPS for Tasselation 15: 23

80 Characters

As expected after the first test with the 70 characters, the tests with 80 characters also showed no significant change in grass height, width or wind intensity.

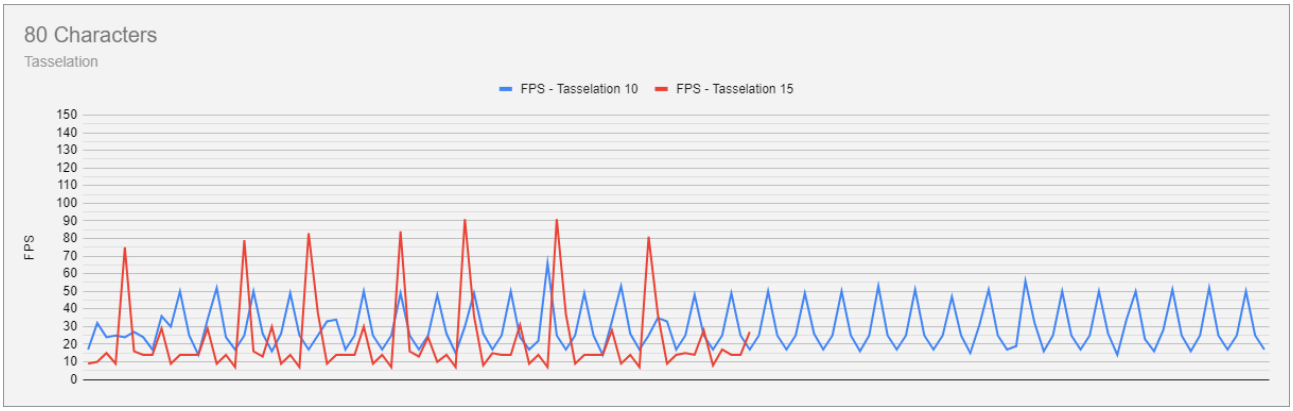


Average FPS for Heigth 1: 63 - Average FPS for Heigth 2: 62



Average FPS for Width 1: 51 - Average FPS for Wind 0.7: 68

In the case of tessellation at 10 and 15, the average FPS values showed that the project was unsustainable, taking into account that the "default" value at which the shader was originally set was with tessellation at 7.



Average FPS for Tasselation 10: 29 - Average FPS for Tasselation 15: 22

Conclusion

It was seen the shader itself is good enough to work properly in a small game project, however, tessellation is a parameter that must be taken into account when applying this shader to certain surfaces, since if the mesh on which it is applied is not correctly formed, the grass areas will be irregular and may cause problems. The number of characters that can interact with this shader is also a key factor, as the user will need to adjust these two parameters so that the performance of the computer does not drop too much if there are many objects with which the user wants this shader to interact.

Discussion

The system used for all of these testing was the one mentioned above. When running these experiments on various hardware, it can be assumed that similar behavior will manifest itself, albeit to differing degrees depending on the system's power. This does not, however, guarantee that running these tests on various hardware would produce behaviors that are identical. Some factors may have a greater or lesser effect on systems employing different hardware.

Git

https://github.com/Helnyr/Advanced-Tools-Analysis/wiki/Analysis