Real-time shader-based rendering of wheat

Kin Liu
Manuel Reinfurt

Figure 1: *End result*

Abstract

The grass is the best.

1 Introduction

Content: Something like "Grass has always been an important factor when simulating nature scenes". Already note some difficulties: "Very high number of vertices/objects"

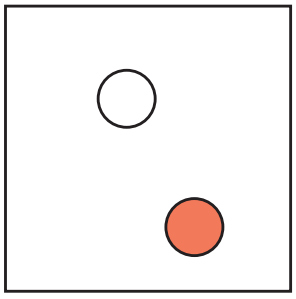

Figure 2: *Sample illustration.*

2 Methods

Explain: General idea of grass, root points, density map. Terrain, height map, Y position displacement. Generating the grass blade with a dynamic number of vertices. Several grass patches in the grid instead of just one patch. Minimizing draw calls.

CPU
The CPU will generate the root points out of the density map and create a single vertex buffer, that contains all roots. It will also slice the full grass grid into smaller patches, which can be controlled through constant buffers on the GPU.

Since we also have static terrain, the root points will already contain the correct displaced Y position.

Vertex shader
The vertex shader is, in essence, a pass-through shader.

Geometry shader
All the hard work is done in the geometry shader. Since the geometry shader gets a single point as an input, it's job is to create a grass blade with a specified number of vertices. The geometry shader will also take care of calculating normals, level of detail, animation.

Pixel shader
To calculate the color, we use the basic Pong BDRF, combined with a texture and a randomized tint.

3 Randomization

The scene looks very artificial if the grass is laid out with exact distances, when each grass blade has the same tint, height or width. However, it is fairly difficult to generate random numbers on the GPU, which is why we have to use certain tricks to have fairly random grass properties. When laying out the root points, we can generate random positions using the CPU. Since these points are given to the geometry shader, and we kow that these positions are random, we can generate a random number between -1 and 1 using the following equation.

$$r = sin(\frac{\pi}{2} * frac(root.x) + \frac{\pi}{2} * frac(root.z)) \qquad (1)$$

There is still a problem, though. Since we now only have one random number, the grass properties will be related to each other. If the grass is a bit higher, it will also be a bit browner.

INSERT non-randomized and randomized picture

4 Animation

Explain: Basic sine animation, Wind

5 Wheat differences

Explain: Flat plane does not work very well.

6 Benchmarks

We kept randomized values to a minimum to avoid variations in the scene and to have a baseline. All tests were mainly done on the following two systems.

System 1 (Desktop)
CPU: Intel Xeon E5-1230v3
RAM: 16GB DDR3-1333
GPU: nVidia GeForce GTX 780

System 2 (Notebook)
CPU: Intel Core I7-4980HQ
RAM: 16GB DDR3-1333
GPU: Intel Iris Pro

Content: Show base FPS without performance improvements, then show level of detail, culling and so on.

7 Conclusion

As you can see, shaders are... best.