Grass Visualization

Tomáš Dubský <xdubsk08@stud.fit.vutbr.cz>

January 4, 2024

1 Introduction

The goal of this project is to implement a real-time graphic demo with lots of grass on a large terrain.

The implemented scene consists of a meadow of size 512×512 meters filled with tall grass. It uses no external textures or meshes, all resources are procedurally generated.

2 Rendering of the scene

The meadow map is divided into tiles of 8×8 meters. At the beginning of a frame, a compute shader is launched (one workgroup of 64 threads per tile). The compute shader performs the following:

1. Test if the tile is inside view frustum, if not exit right away.
2. Calculate the number of grass blades in the tile. The farther the tile is from camera, the fewer blades it gets. Tiles that are nearest to the camera receive 2048 blades (32 blades per square meter), farthest ones receive just 64 blades (1 blade per sq m).
3. Generate geometric information about each blade and store it to a storage buffer. This information contains root position, horizontal rotation, vertical tilt, size, sway strength, color tints, LOD info and a random hash. Sway strength and tilt are controlled by wind which is simulated by a noise that is slowly scrolling across the map.

Once the compute shader finishes, the output shader storage buffer is used for instanced indirect draw of all the blades. Each blades is one instance of 15 vertices. Using the geometric information provided by the above mentioned compute shader, the vertex shader performs the following:

1. Calculate its position along Bezier curve of the blade. The curve is defined by root position, rotation and tilt.

2. Sway the curve/vertex with a sine wave. Amplitude of the wave is based on sway strength and distance from root (the tip sways the most).

3. Offset vertex from the curve's axis to the side of the blade so that the triangles of the blade form a triangle strip.

4. Calculate normal of the vertex. The normal is tilted sideways to give the blade more round look.

The fragments shader shades the blade using Blinn-Phong reflection model.

Finally, dirty ground is rendered below the grass blades. This is done similarly to the grass. It also uses 8×8 meters tiles. If the tile is inside view frustum, it is tessellated based on distance to camera, otherwise it's tessellation level is set to zero which effectively culls it. The texture of dirt is a mixture of noises shaded using Blinn-Phong reflection model. Figure 1 shows the demo in debug mode which shows how frustum culling works.

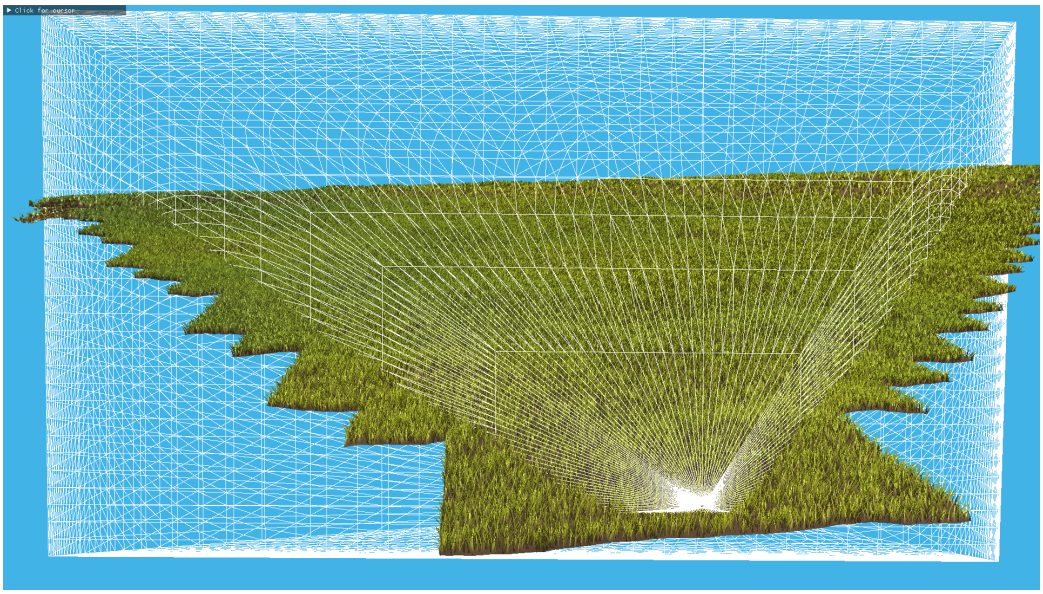

Figure 1: The screenshot shows the demo in debug mode where culling frustum is separated from real viewing frustum. It can be seen that both grass blades and dirt are culled on per-tile basis.

The demo[1] was implemented using C++20 with a small library[2] built on

[1] Available at: `https://github.com/ZADNE/Meadow`

[2] Available at: `https://github.com/ZADNE/RealEngine`

top of Vulkan 1.3 API.

The approach was heavily inspired by Ghost of Tsushima's approach to grass rendering[3].

3 Challenges of grass rendering

The biggest challenge was avoiding alias noise of grass in distance. A number of measures were implemented to suppress it:

- There are fewer blades at distance.
- Blades at distance are much wider.
- Normal vectors of far blades are not so much tilted sideways.
- Shininess of far blades in lower.
- Blades at distance also sway less.

All the measures aim at reducing normal vector variance at distance and thus reducing specular highlight flickering.

Another challenge is finding balance between speed and appearance, in other words adjusting level of detail of grass based on distance to camera. There are many parameters to fine-tune.

[3]Available at: `https://www.youtube.com/watch?v=Ibe1JBF5i5Y`