Advanced Terrain Grass HDRP 12.1

Preview

The associated unitypackage adds *extended* support for HDRP 12.1 by adding compatible shaders and an updated Grass Manager.

The shaders are authored using Shader Graph so you can easily tweak them. Touch Bending as in URP is *not* supported.

Requirements

- Unity 2021.2 or higher (developed using 2021.2.0f1)
- HDRP 12.1

Changes

Installing the package you will update the:

- **GrassManager scripts** to support motion vectors
- **Wind script** which implements easier handling
- **HDRP shaders** which come with improved bending and further optimizations
- some **prefabs** and the **HDRP demo** scene

Updating to HDRP

- In case you update your project to HDRP you have to assign the HDRP shaders manually to your materials.
- Please note that HDRP's transmissive lighting uses *Thickness* not *Translucency*. So you have to invert the translucency channel in your textures and most likely tweak it to fit the used *Diffusion Profile*.
- Simple grass materials with just an Albedo Alpha texture need you to provide a proper "Thickness and Smoothness" texture, assign the "ATGdefaultNTS" texture or uncheck "Enable NTS".
- The HDRP grass shader handles normals differently.

Getting started

After having installed the package you will find a new folder "HDRP" inside the "AdvancedTerrainGrass" folder which contains the HDRP shaders as well as the "Demo HDRP" scene. In order to jump right into it I recommend opening the demo first which comes with a fully configured terrain.

If you open the demo you will notice that the terrain is covered with:

- pretty dark grass and foliage caused by some bug when the terrain engine tries to draw these using GPU instancing (disabling "Light Layers" fixes this).
- some white quads – simply because HDRP has no default grass shader for grass being applied as a simple texture.

This will change when you enter play mode and ATG takes over the rendering.

But before you do so you have to "fix" rendering and add or assign a valid Diffusion Profile:

As the grass and foliage shaders use HDRP's transmissive lighting you have to make sure that the assigned **Diffusion Profile** is added to your **Pipeline Settings**. Do so by editing one of the grass or foliage materials and hit the "Fix" button. Of course you can tweak the profile or assign your own.

> *If the used Diffusion Profile is not added to your Pipeline Settings grass and foliage will get a strange greenish tint.*

Demo HDRP – Details

As mentioned before: Just opening the provided demo you may spot a) some **pretty dark grass** and foliage on the terrain next to b) some weird **white quads** – just because since HDRP 12 the terrain engine may render details using an instanced shader which is fine.

> *a) Dark grass and foliage is caused by some bug in the terrain engine (disabling "Light Layers" fixes this).*
>
> *b) The white quads are details added as simple textures which the terrain engine just does not support when using HDRP.*

Entering play mode ATG will take over grass rendering and a) and b) should be fixed.

Next thing you may notice are some odd **red pixels on the grass**.

> *These just visualize the new "Vertex Fade" feature: Early fading triangles have been overpainted with red. You may edit the albedo texture in photoshop and remove the according layer to get nicely colored grass.*

When entering play mode and **looking at the FPS** you might be shocked: The demo uses settings and models all but highly optimized: The grass model uses really a lot of vertices and the overall density of foliage meshes is pretty high. So take this as a stress test.

Nevertheless: When using ATG and compute I get around 84 FPS (motion vectors enabled) while with ATG disabled FPS are around 60 (motion vectors disabled as these are not supported by the terrain engine). Quite a nice win (tested with a static camera).

Looking at the details you may notice that **bending has been massively reworked**: Old grass jittering has been replaced by a more distinguished turbulence. And foliage does not only support turbulence as well but also supports branch bending along the wind direction.

What's new

Motion Vectors

As with HDRP 12 and Unity 2021 we are now able to render out proper motion vectors when it comes to instanced drawn meshes.

In order to enable motion vectors you have to edit each single prototype in the *Grass Manager* inspector and goto: **Motion Vectors**. Then set it to: **Object**.

Furthermore you will have to enable motion vectors in the materials: Go to the very bottom of the related material inspector and check: **Motion Vector For Vertex Animation.**

> **Please note:** Rendering motion vectors is not for free but rather expensive. So check your FPS and the visual impact after having enabled them.

Improved Bending

The grass and foliage shaders offer some new settings:

Per Instance Variation Wind is sampled from the wind texture which usually gives you well defined, large wind patterns. However if all instances sample this texture as is, the final bending may look odd as it creates *"well defined, large wind patterns"* as well: All instances of a given prefab at a certain location may just bend in full sync.

Here *Per Instance Variation* comes into play as it lets you slightly shift the position used to sample the wind texture based upon a random value generated taking the instance scale into account. In other words: We will keep the incoming global gust but add some local noise on top.

> *Small values like 0.1 should already do the trick here.*

Phase Offset ATG always offsetted the wind sampling coordinate slightly based upon the baked in phase (in vertex color red). *Phase Offset* now lets you determine the scale of this offset.

Branch Bending – along Wind Direction Classic branch bending derived from Crytek's original implementation bends branches only up and down. While this is mostly fine for trees, foliage

may not look as convincing at all. And when it comes to ferns the combination of main bending and classic branch bending might just not give you the desired result. Here *Branch Bending - along Wind Direction* jumps in: It may completely replace *main bending* (which in this case should be set to 0.0). It takes the baked branch bending in vertex color blue into account and also adds some variation based on the baked phase (vertex color red). And – like its name suggests – will push the vertices according to the wind direction.

Have a look into the "PF Fern HDRP" to find out more.

Turbulence Turbulence replaces the old Jitter (Grass) or adds some new detail bending (Foliage) as a small scale analytical noise to the wind animation.

Grass

Turbulence here is driven by the given normal *(direction)*, worldspace position and **Frequency** *(frequency)* and baked main bending *(strength)*. **Frequency** acts as a multiplier to the predefined frequency in shader code.

Foliage

Turbulence here is driven by the given normal *(direction)*, baked branch bending (in vertex color blue) and **Frequency** *(frequency)* – so the waves that are created run along the blue gradient – and baked branch bending *(strength)* – which gets multiplied with baked edge flutter (in vertex color green) according to the **Mask** param: 0 will result in no masking while 1 means full masking by vertex color green.

Have a look into the "PF Fern HDRP" to find out more.

Improved Lighting

Grass may sample a normal map now too.

Improved Optimizations

ATG does not support LODs for various reasons but it comes with other optimizations which increase performance over distance like its Two step culling which fades out a user defined amount of instances early on.

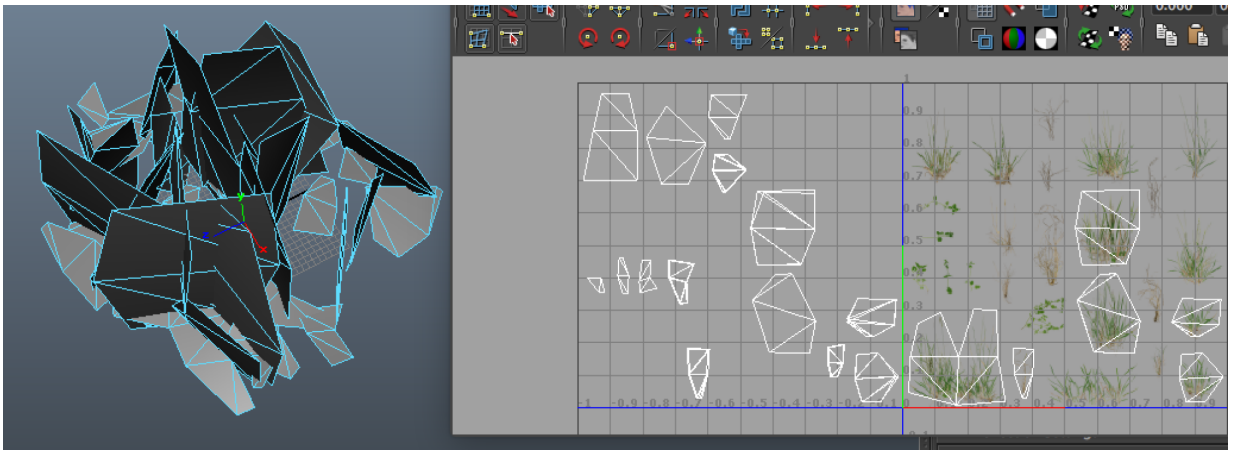
Vertex Fading Latest version for HDRP also adds *Vertex Fading* which allows you to fade out only certain vertices or better: triangles so you can kind of simplify the meshes over distance.If you have a look into the provided HDRP demo the *nettle* and the *grass patch* uses this technique to reduce the amount of quad overdraw: Nettle e.g. fades out the thin trunk while the grass patch fades out all triangles parallel to the ground and even some of its upright oriented grass planes.

Depending on the models' complexity and the amount of early dropped faces I have seen FPS going up by 10 –15%. Under real conditions the savings might be less. But as it is pretty simple to add to your models it is always worth testing it.

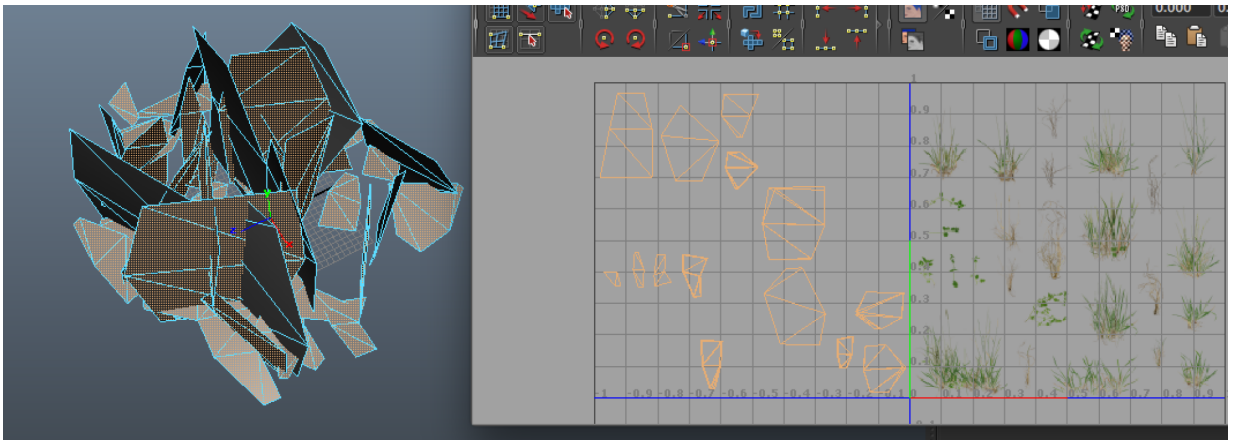
Vertex Fading is calculated within the vertex shader. So here you will gain only a little performance. The big win will be in the fragment shader as it will never run for skipped faces.

In order to mark faces to be dropped early on you have to move their UVs into the negative UV quad so UX.x becomes < 0. The used textures have to be set to tiling in the import settings of course. But this is pretty much all you have to take care of.

UV layout of *MeadowGrassPatch*.



The faces of the *MeadowGrassPatch* which will be dropped by *Vertex Fading*.



Wind Fading Pretty obvious: Fading out wind and dropping the wind animation at a certain distance will improve the performance of the vertex shader and save bandwidth. It will do so even more if you use motion vectors.

Wind fading is driven by the **Wind Fade Distance** which is the distance at which the wind will start fading out and **Wind Fade Range** which specifies the range over which wind will fade out. *Wind Fade Range* is not set in meters but as 1.0 / (range in meters * range in meters) so it equals a pretty small value. A 15m range then would equal 0.0044. In practice a value smaller than 0.001 is not suitable due to floating point precision issues.

> *In order to check the wind fade you may edit the shaders and connect the output "FadeOut" from the "Debug Wind Fade" function and plug it into the "Base Color" slot of the master node. Albedo then will show the calculated wind fade. White = no fade, black = fully faded.*

Improved Wind

Setting up wind hasn't been fun in the past I admit. So this time I introduced some more intuitive params.

First major change: I decoupled turbulence from the turbulence values coming in from the wind zone as it is just too complicated to keep two parameters in sync. Somehow.

Instead you can now specify a **Max Turbulence** and then a curve **Wind To Turbulence** which describes the relation between *Main* (wind strength from the wind zone) to Turbulence as it will land in the global wind texture.

Changed or new parameters:

Wind To Frequency Change The shaders change the frequency of the branch bending and turbulence according to this factor so stronger wind strength may make grass and foliage bend faster.

> *Do not go crazy with this param: A value of 0.25 should be fine.*

Max Turbulence Determines the max turbulence encoded into the global wind texture.

Wind To Turbulence Lets you define a curve which drives how incoming *main* (wind strength) from the wind zone will influence the final *turbulence* encoded into the wind texture.

Base Wind Speed Speed of wind in km per hour at *main* (wind strength from the wind zone) = 1.0 (maps to Speed Layer x = 1.0). So now you have some real world reference.

Size in World Space Determines the area covered by the wind texture before it tiles.

> *Taking Resolution and Size in World Space into account you will be able to get a picture of how many details the wind texture will contain. Or just use the viz prefab to visualize it :)*

Wind Base Tex The package now also contains a texture stored as .EXR giving you 16bit precision. Consider using this in case you think the wind animation suffers from bending artifacts.

Shaders

Channel packing of the textures is as close to the one used in the standard render pipeline as possible. But as the shaders use Shader Graph you can easily change it to whatever you like to support more features, get better compression quality or just make it fit your project.

Please note: Only major differences to the standard shaders will be handled here.

Grass HDRP

Enable NTS When checked the shader will sample the combined normal, thickness and smoothness texture

> **Normal (AG) Thickness (R) Smoothness (B)** HDRP's transmissive lighting uses *Thickness* not *Translucency.* So you have to invert the translucency channel in your textures and most likely tweak it to fit the used *Diffusion Profile*. Normal map will be sampled from A and G like in the foliage shader.
>
> **Normal Scale** Lets you scale the sampled tangent space normal.
>
> **Smoothness Scale** Lets you scale the sampled smoothness. If you unchecked **Enable NTS** the final smoothness will equal *Smoothness Scale*.
>
> **Thickness Scale** Lets you scale the sampled thickness. If you unchecked **Enable NTS** the final thickness will equal *Thickness Scale*.

Wind Sample Radius If set to 0.0 the shader will sample wind at the pivot. Otherwise at the `pivot + VertexPosition * Wind Sample Radius`

Wind LOD (int) lets you specify the LOD level at which the global wind texture gets sampled. Higher LOD levels will create smoother bending.

Per Instance Variation Lets you slightly shift the position used to sample the wind texture based upon a random value generated taking the instance scale into account.

Phase Offset Lets you slightly shift the position used to sample the wind texture based upon the baked phase in vertex color red.

Main Bending Strength of the bending along the wind direction driven by vertex color alpha (Check **Bending Mode Blue** if you store main bending in vertex color blue).

Turbulence Strength of the turbulence (which replaces Jitter) which is driven by the given normal *(direction)*, worldspace position and **Frequency** *(frequency)* and baked main bending *(strength).* **Frequency** acts as a multiplier to the predefined frequency in shader code.

> **Frequency (Multiplier)** Lets you adjust the frequency of the turbulence. The final frequency depends on the wind strength as well and will be tweaked by the wind script.

Wind Fade Distance The distance at which the wind will start fading out.

Wind Fade Range specifies the range over which wind will fade out. *Wind Fade Range* is not set in meters but as 1.0 / (range in meters * range in meters) so it equals a pretty small value. A 15m range then would equal 0.0044. In practice a value smaller than 0.001 is not suitable due to floating point precision issues.

Normal to UpNormal The HDRP grass shader does not tweak the grass' per vertex normals automatically but lets you play around with various settings. You can use the shaders "Normal Mode" to adjust these. And use *Normal to UpNormal* to make the per vertex normal always point upwards.

AO from Bending Amount of ambient occlusion calculated from the baked main bending.

> **Upper Bound** Lets you remap the AO from bending using `smoothstep`: Smaller values here will make AO reaching 1.0 sooner.

Scale Mode XZ only Maps the enum Scale Mode XYZ or XZ into Shader Graph space. If checked the shader will shrink the instances only along the XZ axis.

Diffusion Profile Here you have to assign a regular HDRP diffusion profile.

Foliage HDRP

Normal (GA) Thickness(R) Smoothness(B) HDRP's transmissive lighting uses *Thickness* not *Translucency.* So you have to invert the translucency channel in your textures and most likely tweak it to fit the used *Diffusion Profile*.

> **Normal Scale** Lets you scale the sampled tangent space normal.

Wind Sample Radius If set to 0.0 the shader will sample wind at the pivot. Otherwise at the `pivot + VertexPosition * Wind Sample Radius`

Wind LOD (int) lets you specify the LOD level at which the global wind texture gets sampled. Higher LOD levels will create smoother bending.

Per Instance Variation Lets you slightly shift the position used to sample the wind texture based upon a random value generated taking the instance scale into account.

Phase Offset Lets you slightly shift the position used to sample the wind texture based upon the baked phase in vertex color red.

Stretchiness If 0.0 the shader will keep the original "length" of the vertex position in object space while when set to 1.0 it will stretch the vertices.

Main Bending Strength of the bending along the wind direction driven by vertex color alpha (Check **Bending Mode Blue** if you store main bending in vertex color blue).

Branch Bending Strength of the bending up and down driven by vertex color blue.

> **Along Wind Direction** Strength of the bending along the wind direction driven by vertex color blue. This is close to **Main Bending** but takes phase (vertex color red) into account as well.

Turbulence Strength of the *Turbulence* driven by the given normal *(direction)*, baked branch bending (in vertex color blue) and **Frequency** *(frequency)* – so the waves that are created run along the blue gradient – and baked branch bending *(strength)* – which gets multiplied with baked edge flutter (in vertex color green) according to the **Mask** param: 0 will result in no masking while 1 means full masking by vertex color green.

> **Frequency** Lets you adjust the frequency of the turbulence. The final frequency depends on the wind strength as well and will be tweaked by the wind script.
>
> **Mask** Lets you mask Turbulence by vertex color green.

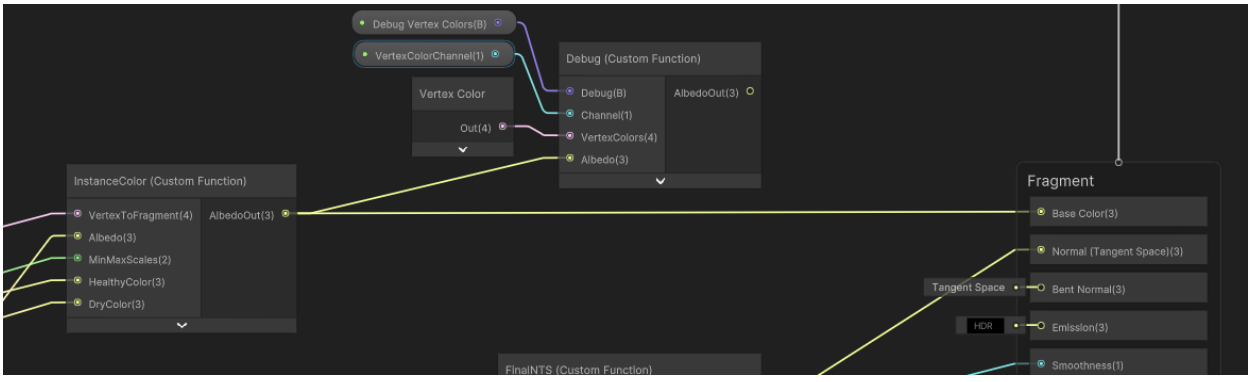
Wind Fade Distance The distance at which the wind will start fading out.

Wind Fade Range specifies the range over which wind will fade out. *Wind Fade Range* is not set in meters but as 1.0 / (range in meters * range in meters) so it equals a pretty small value. A 15m range then would equal 0.0044. In practice a value smaller than 0.001 is not suitable due to floating point precision issues.

Customize and improve Shader Performance

Sample NTS In case you do not want to sample any normal, smoothness and thickness texture, the most performant way would be to disconnect the Normal (Tangent Space) slot in the master node. This should make the shader skip all the tangent to world space calculations and also lowers the needed bandwidth (fewer vertex to fragments interpolators needed).

Debug Vertex Colors In case you do not need it just bypass the Debug node in the Shader Graph. This will skip vertex colors being send from the vertex to the fragment shader which reduces bandwidth usage:



Separate Albedo and Alpha and store it in different textures. Then a 1K albedo RGB texture using DXT1 compression needs 0.7MB as well as a 1K alpha R texture using BC4 compression. In sum this will be the same as using one combined RGBA texture using either DXT5 or BC7 compression (1.3MB) but depth prepass and shadow caster passes would only sample the small 0.7MB alpha texture while the gBuffer pass would only read the 0.7MB rgb texture. This however only makes sense if you use a depth prepass and let your grass cast shadows.