Advanced Terrain Grass URP 12.1+

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

The shaders are authored using Shader Graph so you can easily tweak them.

Requirements

- Unity 2021.3 or higher (developed using 2021.3.21f1)
- URP 12.1.10 or above

Changes

Installing the package you will update the:

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

Updating to URP

- In case you update your project to URP you have to assign the URP shaders manually to your materials.
- The URP grass shader handles normals differently. So you will have to do some adjustments.

Getting started

After having installed the package you will find a new folder “ATG HDRP 12+ SG Shaders” inside the “Shaders” folder as well as the “02 Full Demo URP” scene. In order to jump right into it I recommend opening the demo first which comes with a fully configured terrain.

> **Please note:** The rendered grass in scene view now looks way closer to the final result you will get when ATG takes over rendering because of Unity’s new “GPU instanced” grass. The gap will get even smaller with newer versions of Unity yet ATG will give you faster rendering.

Demo URP – Details

Just opening the provided demo you will spot a white cube with some grass prefabs on top which show some red pixels. These have been added to illustrate the new “vertex fade” feature.

Looking at the details you may notice that **bending has been massively reworked**: Old grass jittering has been replaced by a more distinguished turbulence.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

Shader Graph Shaders

All shaders have been converted to Shader Graph so you can easily tweak them – and I do not have to update them so frequently :)

Rendering Layers

If you look into the inspector of the Grass Manager you will find a new field called “Rendering Layer Mask”. This lets you specify the “Rendering Layer Mask” as uint - which is needed in case you have activated “Light Layers” in your pipeline asset.

> You have to enter a uint here which represents the layer mask you want to use. The uint is treated as a bitmask. Default is 1. If you want to make the grass being lit by “0:Light Layer default” and “1:Light Layer 1” the value must be 3. Only “1:Light Layer 1” would be 2.

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 URP" 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 URP" to find out more.

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 URP 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 demo the *nettle* and the *GrassPatchHighPoly URP* uses this technique to reduce the amount of quad overdraw: Nettle e.g. fades out the thin trunk while the

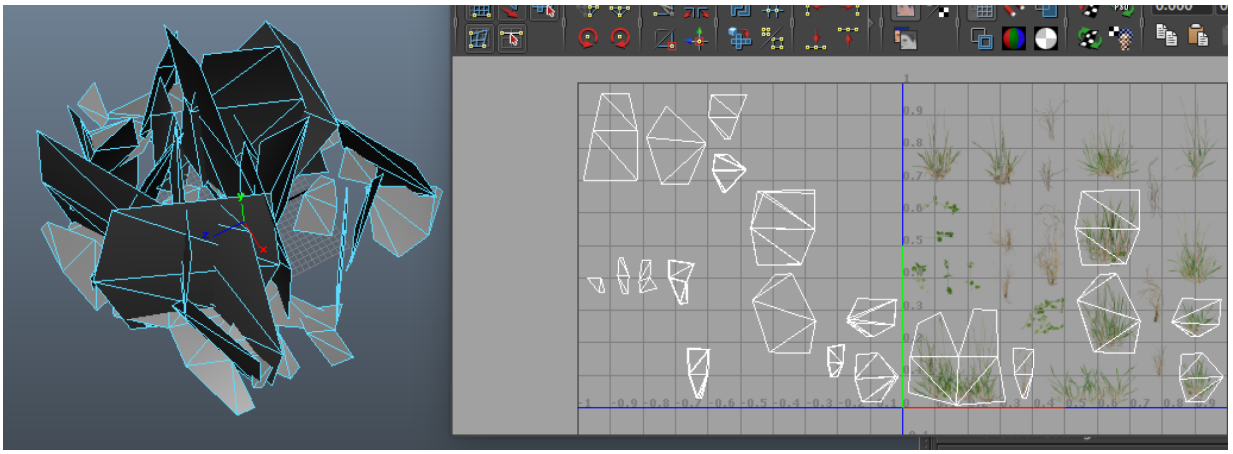
grass patch fades out all triangles parallel to the ground and even some of its upright oriented grass planes.

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.

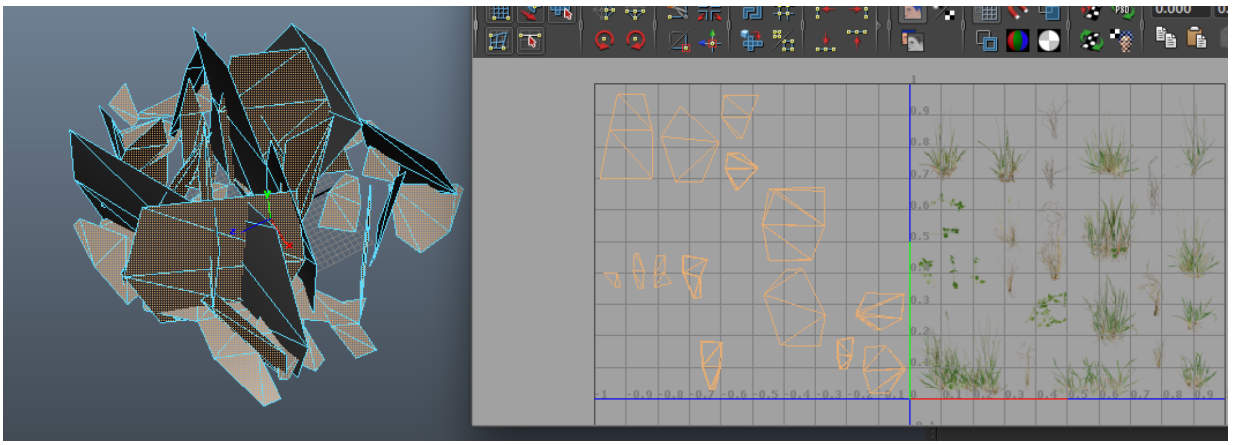
Vertex Fading is calculated within the vertex shader. So here you will gain only a little performance. The big win will be in the fragment shader as it will never run for skipped faces.

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 as less lookups into the wind texture are needed.

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.

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.*

Transmissive Lighting

I removed that from the custom lighting function instead the shaders now use the provided lighting functions and the shaders add transmission to emission on top. This is not as fast as the previous solution but the only way to get it future proof.

Transmission is calculated for the directional light only by default. If you want it from point or spot lights as well you have to check **Additional Lights** in the Transmission category of the material inspector.

Workflow Mode

The grass and foliage shader should be set to *Workflow Mode → Specular* as this gives you the best control over how reflections will look like.

Grass SG URP

> **Please note:** The currently provided Shader Graph does not support texture arrays unlike the previous HLSL version. If you need this feature you will have to implement it yourself.

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.

Enable VFace if checked the shader will flip the per vertex normals. Useful or needed if you do not smooth the normals.

Normal to up Normal The 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.

Foliage SH URP

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

Custom bending params ATG defined its own format for how bending information must be stored in vertex colors. The latest shaders let you create your own BendingParams and then pass them to the vertex shader. So you can easily define main bending using PositionOS.y or grab it from UVs.

Normals In case you do not want to sample any normal 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:

Advanced Terrain Grass HDRP 12.1+