Advanced Foliage Shaders 5.076

Hi there and welcome to the documentation of the Advanced Foliage Shaders v.5.076.

This package was built to improve the visual impact of almost all vegetation within your project while speeding up rendering at the same time.

For these reasons it does not only offer advanced lighting functions, unified and fully controllable procedural wind animations and features such as touch bending but also GPU instancing on certain shaders and scripts which will combine hundreds of single plants and game objects to just a single mesh (at runtime or within the editor).

The following introduction **AFS in a nutshell** gives you a brief overview of the basic features and will point you towards other chapters of the documentation which provide in depth information.

If you want to get started right away please head for the **Quickstart Guide.**

Learn more about the included **Demos**.

If you have upgraded from version 4 you might be interested in the chapter **Changelog and new features.**

In case you encounter any problem please have a look at the section **Known Issues and FAQ**.

If you want to use AFS along with third party products such as VegetationStudio or Critias Foliage please have a look **here**.

AFS and Unity >= 5.6.

Unity 5.6. forces you to manually enable instancing per material.
While this is a bit nasty for plants and foliage but possible you simply can't do as far as trees are concerned because the "optimized materials" are not editable.

So you may either create and assign your own "optimized" materials (as the afs optimized shaders are not "hidden") or use the updated "treeafsproperty" script which you can get here.

You have also to manually disable "Bake Light Probes For Trees" in the terrain settings in case you use the AFS tree shaders.

As Unity 5.6. has changed the way matrices are handled once again, the built in feature “preserve length” might lead to flickering when using forward rendering and spot or point lights. It can be disabled/enabled in the wind settings of the Setup script.

AFS in a [rather big] **nutshell**

The shaders

The package ships with dedicated shaders for foliage, trees created using the built in tree creator and manually placed grass as well as shaders which replace the built in billboard shader, the built in grass shader for the terrain and the VertexLit shader which is also used by the terrain engine.

The foliage shader supports advanced procedural wind animation, physically based shading including deferred translucency, physically based wetness and GPU instancing on manually modelled geometry. New since 5.02: LOD cross fading is supported
Find out more >

In case you enable touch bending objects using the foliage shader might even interact with your player or any enemy in your scene.
Find out more >

The tree creator shaders replace the built in tree creator shaders and support advanced procedural wind animation, physically based shading including deferred translucency, physically based wetness [beta], GPU instancing and give you pretty smooth transitions between mesh instances and billboards.
Find out more >

The grass shader supports manually placed grass on any kind of geometry, advanced procedural wind animation, deferred translucency and picks up lighting from the underlying geometry in order to make grass truly grounded. Simple wetness is in beta.
Find out more >

The billboard shader [overwrites the built in one] offers camera aligned billboards. Due to the gaining popularity of depth based image effects such as volumetric lighting it renders using alpha testing instead of alpha blending by default making billboards properly write to the depth buffer and play nicely with such kind of image effects.
Find out more >

The foliage shader for the terrain engine [overwrites the built in VertexLit shader] lets you place normal mapped, physically based shaded and procedurally animated foliage even using the terrain engine.
Find out more >

The grass shader for the terrain engine [overwrites the built in Waving Grass shader] supports advanced and globally controlled, unified bending and deferred translucency. Simple wetness is in beta.
Find out more >

Authoring features

The Foliage tool lets you adjust the bending of your manually modelled models right within unity. Find out more >

The Grass Model Postprocessor makes it an ease to bake bending information into your grass meshes by simply editing the file name.
Find out more >

The Geometry Brush [free add on] lets you simply paint foliage and grass on any arbitrary geometry as long as is has a collider.
Find out more >

Unified wind which is driven by a directional Wind Zone and the “Setup Afs” script will sync wind and bending between trees, foliage and grass.
Find out more >

Performance features

As rendering hundreds or even thousands of instances will most likely kill your frame rate the package contains scripts and features which will help you reducing the number of draw calls.

GPU instancing support for foliage and trees.

The combine children script lets you combine a vast amount of objects and meshes into just one resulting in one single draw call.
Find out more >

Convert trees authored using the built in tree creator into “simple trees” and render them with just a single draw call instead of two.
Find out more >

Add foliage as trees to the terrain using the “simple trees” shader and benefit from the built in billboarding feature which lets you draw hundreds of billboarded instances at far viewing distances with just a single draw call.
Find out more >

Easily set up layer based culling using the “Setup Afs” script and cull small objects like grass early.
Find out more >

Reposition terrain trees and place them on top of any geometry while still benefitting from billboarding.

Find out more >

Table of Content

Lighting Features and Annotations

Ambient Lighting

AFS supports unity 5's ambient lighting even on billboards which will give you a much more lively shading than legacy ambient lighting (which used to be a single color) and also supports built in ambient specular reflections—at least as far as the foliage and tree shaders are concerned.

Light Mapping

Light mapping is supported by the grass and the foliage shaders.

The tree shaders do not support light mapping as they use the 2nd UV set to store bending values—however trees placed within the terrain engine will pick up a single color value from the terrain's light map.

The grass and foliage shaders support all light mapping modes which are:

- Non Directional
- Directional
- Directional Specular

If you want to use light maps on manually placed plants which are/get combined using the "Combine Children AFS" script you have to combine those statically before baking the light maps. Please make sure that "Create unique UV2" is checked so that the combined mesh will have unique second UVs for all triangles. Doing so will allow you to even light map single sided grass.

Linear Colorspace

Linear colorspace is like a key to physically based rendering. But using the linear colorspace comes with some special and very important demands of which may be the most important is:

Please note: All textures which are not diffuse albedo textures have to bypass the sRGB sampling! I really can not overemphasize how important this point is—as i stepped into this trap myself several times ;-)

The foliage shaders commonly use some combined "normal, translucency, roughness/smoothness" textures. Those textures have to bypass the sRGB sampling under any circumstance: Just select the texture in the "Project" tab, set "Texture Type" to "Advanced" and check "Bypass sRGB Sampling" in the inspector.

The tree shaders use even 2 of those combined textures: Normal map(GA) Spec(R) and Trans(RGB) Gloss(A): Both should bypass sRGB sampling as well.

So in case you get some strange bump mapping or lighting (especially noticeable if the transitions between billboards and mesh trees do not look smooth) this is probably caused by the fact that the combined textures do not bypass sRGB sampling.

GI, Light Probes and Instancing

One of the new features in AFS 5 is the support of GPU Instancing which allows you to draw many "identical" game objects with only a few draw calls. Objects which shall be drawn

instanced have to share the same mesh and the same material – but may differ in scale rotation and position of course: So drawing a cluster of 50 banana prefabs just should work fine.
However some of Unity's lighting features make break instancing which – in particular – are GI and Light probes:
As soon as you add a light probe to your scene instancing on plants using the foliage shader will be disabled (instancing on trees placed using the terrain engine however will still work).
This is caused by the fact that all rendered by default are set to use and blend light probes – so each instance would receive unique light probe information based on its position.
In order to get around this you may either:

- deactivate *Blend Probes* in the mesh renderer component of the game objects which shall be drawn instanced.
- or make sure that all game objects within a certain area which shall be drawn instanced use the same *Anchor Override*. THis worked a while back but does not work in Unity > 5.5 :(

Basic Requirements

Models and Textures

Before you can use your own models with the advanced foliage shaders you have to make sure that your geometry and textures fit the requirements. Please have a look at the **Model and Texture Guide** to find out more.

Scripts

You always have to add the "Setup Afs" script to your scene as it controls a lot of the parameters of the shaders like wind and rain.

Further features like touch bending uses their own additionally scripts.

Compatibility

APIs

DX9 and OpenGL are only partly supported as both APIs do not support instancing. So bending will look slightly different (your models will be more distorted) and you will not benefit from instancing of course.
I highly recommend to use DX11 or OpenGLCore instead.
In case you want or have to use DX9 or legacy OpenGL you will have to adjust the *Wave Size* for the foliage shader in the *AFS setup script*: Instead of using rather small values you will have to use rather large values.

AFS and other packages

The advanced foliage shaders should be compatible with almost all other packages—however and as it overwrites some built in shaders— there may be some conflicts with packages that do the same.

Built in shaders that get overwritten by AFS are:

- WavingGrass.shader
- VertexLit.shader
- BillboardTree.shader

Please make sure that no other package overwrites these shaders as well (like e.g. Skyshop).

AFS also adds some custom shader passes to the built in "Camera-DepthNormalTexture" shader to make depth based image effects work correctly.

So it ships with a modified version which is located at: "Advanced Foliage Shader" -> "Resources" -> "Camera-DepthNormalTexture".

Please note: You must not move the shader as it relies on several includes.

Please note: Make sure that your project does not contain any other "Camera-DepthNormalTexture" shader as this might overwrite the one that ships with AFS. In case you use a package with also needs to modify this shader you will have to manually merge both.

AFS and the Custom Tree Importer

The advanced tree shaders are prepared to work with trees imported using the "Custom Tree Importer" package from the asset store and may even support tumbling.

Simply assign the “Afs Tree Creator Leaves Optimized CTI” shader to the leaf material.
The bark material can just use the regular “Afs Tree Creator Bark Optimized” shader – but you have to make sure that “Extra Leaf Bending” is set to 0.0.

Using the “Afs Tree Creator shaders with CTI trees will make you benefit from advanced billboards blending and fading in shadows.

Alternatively you may edit the standard Afs tree shaders manually which would need you to tweak the following shaders:

- Resources/Camera-DepthNormalTexture.shader
- AfsTreeCreatorBarkOptimized.shader
- AfsTreeCreatorLeavesOptimized.shader

Please find and comment:
```
#define XLEAFBENDING
```

Then uncomment:
```
//#define LEAFTUMBLING
```

AFS and Lux, Alloy or UBER

In order to implement deferred translucent lighting AFS ships with its own deferred light and deferred reflection shaders.

As Lux, Alloy and UBER all bring their own deferred lighting (and reflection) shaders you may have to enable support for their specific deferred translucent lighting solution.
Please note: Depending on which deferred lighting solution you use, you will get slightly different results. But the basic translucent lighting just should work.

Do so by editing the “AfsPBSLighting.cginc” and simply uncomment either:
```
// #define USEALLOY
```
or:

```
// #define USEUBER
```
Then reimport the AFS folder.

In case you use UBER please note that AFS translucency always uses “Translucency Setup 1” defined in the “UBER_Deferred Params” script. Recommended Settings are:

Strength Value between 8 and 10 should be fine.

Scattering I would keep it rather low e.g. 0.1.

Spot Exponent A value around 10 should match Lux’s lighting.

NdotL Reduction Keep it rather low <= 0.5 to get nicely backlit grass.

AFS and Critias Foliage

Since version 5.06 AFS offers a basic support for the Critias Foliage system which lets you place a vast amount of foliage manually even at custom geometry without having to use the Combine Children script.

Currently supported are the base Foliage shader and manually places grass using the AfsGrassShader (Critias instanced).

Touch Bending and Tree Creator Trees are not supported by the Critias Foliage system right now.

The Demos

AFS ships with various demos which show you how to setup and use certain features.
By default all cameras used in these scenes are set to render in deferred. So in order to get proper visual results you first have to e to make sure that your project is setup properly.

Prepare your project

- Before starting please make sure that your project is set to use the **linear color space** in *Edit -> Project Settings -> Player.*
- In case your camera uses **deferred rendering** you also have to assign the **Afs deferred lighting shaders** in *Edit -> Project Settings -> Graphics*:
    - Under the *Built-in shader settings* change *Deferred* to *custom*, then assign the *Afs_Internal-DeferredShading* shader.
    - Also change *Deferred Reflections* to *Custom* and assign the *Afs_Internal-DeferredReflections* shader.
- In case your camera uses **forward rendering** and you have any depth based image effects applied (like ssao) you have to assign the **Afs_Camera-DepthNormalTexture** shader in *Edit -> Project Settings -> Graphics*:
    - Under the *Built-in shader settings* change *Depth Normals* to *custom*, then assign the *Afs_Camera-DepthNormalTexture* shader.
- If you want to use AFS along with shader packages such as **Alloy**, **UBER** or **Lux** please have a look here.

- You have to enable **dynamic batching** in *Edit -> Project Settings -> Player* to take advantage of **GPU instancing**.
- As some models used in the demos need included **postprocessing scripts** to run over the meshes you should simply re-import the entire “Demos” folder in order to make sure that all models are imported the way they are intended to be.

Foliage Shader Overview Demo As you might expect this scene shows most features from manually placed foliage and grass, trees using the advanced tree shaders to foliage being affected by GI.

Touch Bending Demo demonstrates how to set up touch bending for your player and enemies.

Tree Demo lets you explore the difference between the various tree shaders.

Wind Demo demonstrates the Leaf bending feature of the foliage shader.

Light Mapping and GI Foliage Demo demonstrates how to setup light probes or bake GI.

CTI Trees Demo shows how manually modeled trees imported using the Custom Tree Importer will look like when using the included tree shaders.

LOD Groups Demo shows a simple example of you may use LOD groups to place foliage on the terrain as “trees” which support casting shadows and smooth LOD blending.

Cast contains all included plant prefabs and helps you to set up the bending per prefab, the material bending parameters and the wind parameters of the AFS Setup script so that all plants react properly to any change of the wind.
Use this scene as your Driving Range to set everything up.

Quickstart Guide

This simple tutorial will guide you through some basic steps such as:

- manually adding plant prefabs to your scene.
- using the Combine Children script.
- adding manually placed grass.
- adding trees.
- and adjusting the global wind.

But before you can start you have to make sure that your project is setup properly:

- Please make sure that your project is set to use the **linear color space** in *Edit -> Project Settings -> Player*.
- In case your camera uses **deferred rendering** you also have to assign the **Afs deferred lighting shaders** in *Edit -> Project Settings -> Graphics*:
    - Under *Built-in shader settings* change *Deferred* to *custom*, then assign the *Afs_Internal-DeferredShading* shader.
    - Also change *Deferred Reflections* to *Custom* and assign the *Afs_Internal-DeferredReflections* shader.
- In case your camera uses **forward rendering** and you have any depth based image effects applied (like ssao) you have to assign the **Afs_Camera-DepthNormalTexture** shader in *Edit -> Project Settings -> Graphics*:

    - Under the *Built-in shader settings* change *Depth Normals* to *custom*, then assign the *Afs_Camera-DepthNormalTexture* shader.
- If you want to use AFS along with shader packages such as **Alloy**, **UBER** or **Lux** please have a look here.
- You have to enable **dynamic batching** in *Edit -> Project Settings -> Player* to take advantage of **GPU instancing**.

If everything matches the needs of the Afs shaders we can finally start.

1. Create a new scene and drag the "- - AFS Setup Prefab" into it. The prefab contains the "Setup Afs" script which is mandatory as it drives all features of the shaders and a standard wind zone which acts as "main wind generator".
2. Create a new terrain.

Adding the first plant

3. Grab a foliage prefab from the project tab e.g. the "Banana Atlas SingleSided UV4 Pre" Please note: As soon as the Banana appears in the scene view you will spot some gizmos and handles which belong to the "AFS Foliage Tool": Simply ignore these for now.
4. Duplicate, move, rotate and scale the banana several times.
5. Ensure that your camera views your bananas and hit play.
   You will see some beautifully bending bananas—all in their own and slightly different phase.
6. If you open the frame debugger you will notice that all bananas will get drawn within a single or just a few instanced passes due to GPU instancing (which will only happen if you use DX11 and above, OpenGL 4.1 and above, Xbox One or PS4 – mobile platforms are not supported by Unity).

Adding the "CombineChildrenAFS" script

7. Now let's create a new empty game object, name it "FoliagePatch" and parent all bananas under this game object.
8. Select the empty "FoliagePatch" and attach the "CombineChildrenAFS" script. Just leave all values at default and hit play.
   In case you are curious and want to find out more about the script right now you will find in depth information here.
9. While still in play mode you will notice that all bananas have been gone in the hierarchy. Instead only the parent game object is left which now has a mesh renderer component and a dynamically combined mesh attached to.
   Bending might look a bit different from before caused by the fact that now the foliage shader does not know anything about single bananas but only gets one single huge mesh.
10. As soon as you quit play mode the bananas will be back in the hierarchy.

Adding a second specie

11. Now let's add a second specie e.g. "Fern Atlas SingleSided UV4 Pre" by dragging it to the scene view. Then duplicate, move, rotate and scale the fern several times.
12. Entering play mode and using the frame debugger you will see that all ferns should be drawn within a single or just a few instanced passes.
13. Quit play mode, then parent the ferns under the "FoliagePatch" and hit play again.
14. Now ferns and bananas will be batched into one mesh and drawn within a single pass due to the fact that both plants uses the same "Foliage Atlas Single Sided UV4" material. This requires that both prefabs have bending information stored in UV4 and vertex colors.

    Both methods—GPU instanced game objects as well as combined meshes—should already give you a great performance improvement compared to simple game objects which can not be batched dynamically due to their complexity or statically batched game objects as latter consume a lot of memory and CPU power.
    Which method is best for your project is related to many different factors like size and complexity of your level and will be discussed in another chapter.
    But what is about manually placed grass?

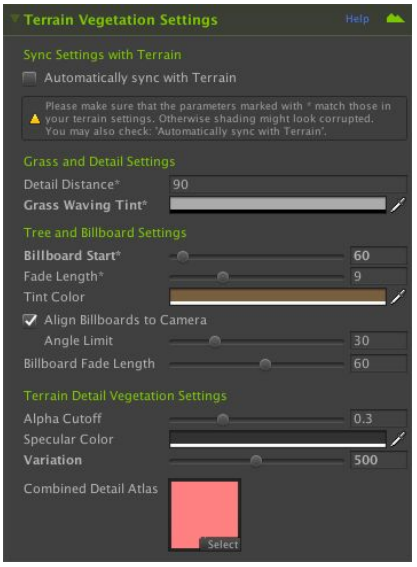
    Adding manually placed Grass
    Manually placed grass always needs the "CombineChildrenAFS" script as it will not only combine all child meshes, save a lot of draw calls and reduce overhead by getting rid of hundreds of game objects in the hierarchy but also bake the normals of the underlying geometry into the combined mesh so lighting will perfectly fit to that of the environment.
15. So grab a grass prefab from the project tab e.g. "Grass SingleSided Pre" and drag it to the scene view. Then duplicate, move, rotate and scale it several times.
16. As soon as you hit play you will notice that the manually added grass meshes change their color and lighting. This is caused by the fact that the grass shader differentiates between edit and play mode: In edit mode lighting is controlled by the orientation of the grass mesh whereas in play mode the grass shader expects the normals of the underlying geometry to be baked into the grass mesh. As these are missing lighting in play mode will most likely look odd.
17. Let's fix this by placing the grass prefabs under our "FoliageCluster" game object.
18. Now the lighting on grass should look OK-ish in edit mode and pretty fine in play mode.
19. As grass uses a different material than banana and fern you will notice that under our game object the script has generated 2 new game objects: one for fern and banana using the foliage shader and another one for the grass using the grass shader.
    You could of course also create a new game object, attach the "CombineChildrenAFS" script and parent the grass under this new one. This would allow you to control if grass should cast shadows independently from fern and bananas.

    Adding Grass to the Terrain
20. Add the same grass prefab to your terrain and start painting some grass. When entering play mode please notice how smoothly manually placed grass and grass on the terrain fade—unless grass on the terrain does not fit in size or uses crazy (default...)

dry and healthy colors.

Adding Trees

21. Finally let's add some trees to the terrain by assigning e.g. the prefab "Conifer [TreeCreator]" to the terrain and painting a few on top of the terrain.
    This prefab uses the Afs Tree Creator shaders. And as by importing the Afs package you also imported the Afs Billboard shader (which automatically overwrites the built in one) trees might behave a little bit strange.
22. In order to make trees work properly we have to set up the tree shaders and feed in some global parameters.
    Do so by selecting the "- - AFS Setup Prefab" in the hierarchy and look for the **Setup Afs script** in the inspector which is attached to the prefab.
23. Open its **Terrain Vegetation Settings** foldout as here we have to tweak the settings of **Billboard Start** and **Fade Length** to make them match the corresponding settings on the terrain.
    The easiest way to do so is to check **Automatically sync with Terrain** which will make a new slot named **Specify Terrain** appear.
    As soon as you drag your terrain from the hierarchy into this slot, the corresponding values will be grabbed from the terrain and the GUI disabled



Adjusting wind

24. As you will have noticed during the previous step: The Setup Afs script offers a lot of different foldouts, tweaks and controls related to various aspects of vegetation shading like its **Wind Settings** foldout which – in combination with the **Wind Zone** component – drives the final wind which gets applied to all different kinds of vegetation be it trees, foliage or grass.
    ***Tip:** When setting up the wind it is pretty usefull to turn on **Animated Materials** in the Scene view tab as you will get a live preview and do not have to change to play mode. Unfortunately however grass isn't animated unless the applications is playing.*
25. Adjust the **wind direction** by rotating "- - AFS Setup Prefab" and watch the immediate response in the scene view.
26. Switch to the wind zone component in the inspector and explore its properties:
27. Set all wind zone properties to 0.0 – except for Main Wind.
28. Enter play mode (in case you haven't turned on **Animated Materials**), then change the **Main wind** and you will see how it drives the main or primary bending of plants and trees along their y axis.
    If all other parameters were set to 0.0 changing main wind would cause a simple bending which *usually* did not change over time (the AFS tree creator and foliage shaders however always add a bit of variation).
    In order to add time we have to raise **Pulse Magnitude** and **Pulse Frequency**.
    Let's add some more variation by raising **Turbulence** as well which drives the

secondary bending mainly affecting the branches moving them up and down.
Please note: Do not change **Pulse Frequency** in play mode. Unity's tree component does not handle this (bending jitters and glitches) and so there is no need the foliage shader should do.
Please note: You must not **change the wind direction** in play mode by a large amount of degrees within a short period of time as otherwise **grass will go crazy**. In case you want to change the wind direction you will have to do it smoothly over a long period of time, sorry.

Now that you are familiar with the wind zone's properties let's have a look into the **Setup Afs script** and open its **Wind Settings** foldout

29. This foldout contain three subsections: one for each specie. Each subsection contains multipliers for the wind zone's **Main** and **Turbulence** properties which are named **Main Strength**, **Turbulence Strength** or simply **Strength**. These let you easily adjust the bending for e.g. all trees independently from grass or foliage.
For an in depth description of all other properties please have a look at the chapter Wind Settings, read the tool tips (which only show up if you are not in play mode) or simply play around with them.
Please note: Properties marked with "*" shall not be changed in play mode as they might cause glitches. In fact i would recommend to use the "Wind settings" only in editor to fine tune your wind. While your game is running changing the wind zone's properties only should give you enough control.

Now that you have added some plants, grass and trees, had a first look into the Setup Afs and Combine Children script it might be time to learn how to prepare, import and edit custom models. You will find all relevant information in the section **Model and Texture Guide**.

As placing hundreds of blades of grass manually is quite cumbersome the package ships with the **Geometry brush** as free addon, which lets you simply paint prefabs into the scene. So it is worth having a look into it.

If you are looking for how to setup touch bending you will find the needed information in the chapter **Touch Bending**

AFS Setup Prefab

When using the Advanced Foliage Shaders all trees, plants and even grass are driven by one global directional wind zone. Yes, even grass placed on the terrain will react to any change of the wind direction or speed!

This wind zone is accompanied by the "Setup Afs" script – both bundled in the "AFS Setup Prefab".

The "Setup Afs" script lets you fine tune some parameters to make the different species just fit better and passes all other needed parameters to the shaders. It must be part of any scene you would like to use the shaders in.

Wind Zone

Unity's manual does not really contain a lot of information about wind zones. And as foliage and grass are not affected by radial wind zones the global wind zone has to be directional which reduces the info we are given to what you will find below (trees however might still be affected by additional spherical wind zone):

Main The primary wind force. Produces a softly changing wind pressure.

Turbulence The turbulence wind force. Produces a rapidly changing wind pressure.

Pulse Magnitude Defines how much the wind changes over time.

Pulse Frequency Defines the frequency of the wind changes.

Setup Afs Script

Wind Settings

The wind settings foldout lets you fine tune the influence of the wind zone on all different shaders that come with the package.

Preserve Length If enabled the foliage shader will reduce distortions caused by the wind animation. However this might lead to flickering when using forward rendering and spot or point lights. It should always be fine to enable it when using deferred rendering though.

Wind Settings for Foliage Shaders

Main Strength Factor for the "Main" wind strength as defined in the wind zone.

Turbulence Strength Factor for "Turbulence" as defined in the wind zone.

Leaf Turbulence* Factor the original frequency of the secondary bending gets multiplied with. It determines the max frequency leaves might bend at when affected by strong winds. Leaf Turbulence gets multiplied with the corresponding Leaf Turbulence property of the material.
***Please note:** Do not change this parameter in play mode as it might cause glitches.

Wave Size* The shader adds some variation to the bending taking the object's position in world space and the "Wave Size" parameter into account. Smaller wave sizes will add more variety to a given area. "Wave Size" is most important in case you have neither enabled instancing in the material inspector nor combined children and baked pivots so the shader expects dynamically batched plants and just a soup of vertices.
***Please note:** Do not change this parameter in play mode as it might cause glitches.

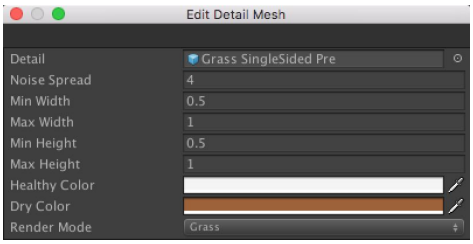
Wind Settings for Grass Shaders

Strength factor for the "Main" wind strength as defined in the wind zone.

Speed corresponds to the original "Speed" parameter in the terrain settings but gets automatically adjusted according to the given wind strength.

Wave Size* corresponds to the original "Size" parameter of the terrain engine: The size of the "ripples" on grassy areas as the wind blows over them. Higher values will produce denser waves.
***Please note:** Do not change this parameter in play mode as it might cause glitches.
***Tip:** Debug Wave Size by setting "Grass Tint" to e.g. red.*

Turbulence* lets you break up the more or less uniform shape of the wind waves. It is calculated per grass “instance” based on vertex color red.



So when setting up your grass meshes using the “Edit Detail Mesh” dialog of the terrain engine you should make sure that you apply proper “Dry” and “Healthy” colors as well as a rather high “Noise Spread”.
***Please note:** Do not change this parameter in play mode as it might cause glitches.

Jitter gives you a second wind animation on top which breaks up uniformity even more. It needs Turbulence to be set to > 0.0 to work properly.

Wind Settings for Tree Creator shaders

In case you want to mix Speedtrees with tree creator trees you will probably come across the problem that Speedtrees react much more picky to the wind applied using a built in wind zone. So you can use the following parameters to adjust wind bending of all trees using the tree creator shaders.

Main Strength Factor for the “Main” wind strength as defined in the wind zone.

Turbulence Strength Factor for “Turbulence” as defined in the wind zone.

Terrain Vegetation Settings

This foldout lets you configure trees, foliage and grass placed using the terrain engine and syncs them with manually placed objects.

Automatically sync with Terrain If checked the script will automatically grab the needed values from the terrain assigned in:
Specify Terrain The terrain the script will grab the needed values from.

Grass and Detail Settings

Detail Distance The distance (from camera) beyond which details like grass will be culled. Make it fit "Detail Distance" in your “Terrain Settings” in order to perfectly sync manually placed grass.

Grass Waving Tint Color that will be applied to the manually placed grass. Make it fit "Grass Waving Tint" in your “Terrain Settings” in order to perfectly sync manually placed grass.

Tree and Billboard Settings

Billboard Start The distance (from camera) at which 3D tree objects will be replaced by billboard images. Must fit "Billboard Start" in your “Terrain Settings”.

Fade Length Distance over which trees will transition between 3D objects and billboards which must fit the "Billboard Start" in your “Terrain Settings”.
If a tree has fully transitioned to 3D shadows casted by the tree, specular lighting and translucency will fade in over a distance which equals the specified “Fade Length”.
Find out more >

Tint Color Usually trees are simply tinted with black if you set the “Color Variation” > 0.0 while painting trees. You may replace this color by any one you like using this color field.

Align Billboards to Camera Check this to get camera aligned billboards. If unchecked you will get back the "always upright oriented" billboards. All enhanced fading features for mesh trees (wind, shadows, light) still will be present.
Please note: Using perspectively aligned billboards might look pretty strange within the editor as the script has to pass some parameters from the "current" camera to the shader. In order to give you a correct preview in the "game view" the script will only look for a camera tagged as "MainCamera". All other cameras will be ignored. So if you need a second camera (e. g. for rendering reflections) you will have to update the shader parameters according to the second camera before rendering. Look for the block:

```
Shader.SetGlobalVector("_AfsBillboardCameraForward", Vector4(
CameraForward.x, CameraForward.y, CameraForward.z, 1));
```

to find out how to set up the needed parameters correctly.

Angle Limit Defines the angle (when viewed from above) the billboard shader will fade to the "classical" upright oriented billboards in order to reduce distortion artifacts. The standard value is "30".

Billboard Fadeout Length Instead of simply culling the billboards beyond the "Tree distance" specified in the terrain settings you may define a length over which the billboards will be smoothly faded out.
Please note: This feature needs the “AfsBillboardTree Transparent” shader to be be activated. Do so by removing the “AfsBillboardTree CutOut v5.shader” from the project (or renaming it to “._shader”) and renaming “AfsBillboardTree Transparent._shader” to “AfsBillboardTree Transparent.shader”. You will have to restart Unity to make it pick up the new shader.

Terrain Detail Vegetation Settings

Legacy Bending Check this in case all foliage meshes using the vertexLit shader are prepared for legacy bending which means: Vertex color blue stores primary and secondary bending while alpha stores ambient occlusion.
Uncheck this if your foliage meshes are prepared to use the new or regular vertex color bending which stores secondary bending in vertex color blue and primary bending in vertex color alpha.

Color Variation Lets you you define a color which gets multiplied on top of the albedo and varies from instance to instance. Alpha defines the maximum strength.

Alpha Cutoff Cut off value for all models placed as "detail mesh" using the built in terrain engine.

Combined Detail Atlas Assign the combined texture atlas containing the normal, translucency and smoothness maps.
Please note: If you do not assign a custom texture here, the script will assign a default one to make all plants look some kind of OK-ish.
Please note: This texture has to be imported using *Advanced -> Bypass sRGB Sampling* checked. You most likely also want it to be sampled using *Filter Mode = Trilinear* as it contains a normal map.

Specular Color Specular Color which gets applied to all models placed as “detail mesh” using the “VertexLit” shader.

Translucency Strength lets you scale translucency for all meshes placed within the terrain engine using the “VertexLit” shader.

View Dependency determines when the translucent lighting effect will kick in depending on the view angle (for all meshes placed within the terrain engine using the “VertexLit” shader): Lower values will make translucent lighting appear already at rather flat viewing angles while high values will make it appear only if you look directly towards the sun. Values between 0.7 – 0.8 should be fine in case you want some kind of traditional thin layer translucency.

Backface Smoothness lets you adjust the smoothness on back faces of meshes placed within the terrain engine using the “VertexLit” shader . This feature only works with single sided leaf geometry.

Horizon Fade lets you suppress “wrong” ambient specular highlights which might be caused by “strong” normals from the normal map which might make the reflection vector end up pointing behind the surface being rendered. For further details please have a look at: http://marmosetco.tumblr.com/post/81245981087

Leaf Turbulence* Global factor for all plants using the vertex lit shader. Increasing Leaf Turbulence will make those parts of the plants which are affected by secondary bending sway faster if the wind blows stronger.
***Please note:** Do not change this parameter in play mode as it might cause glitches.

Variation Multiplier which controls the amount of variation in bending. Larger terrains needs larger values. As a rule of thumb i would recommend to simply take your terrain width.

Grass Settings

Min Smoothness Base Smoothness of grass masked by translucency as sampled from the *Combined Detail Atlas*.

Min Smoothness Max Smoothness of wet grass masked by translucency as sampled from the *Combined Detail Atlas*.

Rain Settings

Rain Amount Lets you fade in physically based wetness on models using the foliage and tree (beta) shaders.

The shader will darken albedo, raise smoothness and tweak the specular color towards the physically properties of water taking the rain amount, the world normal up direction of the given pixel as well as its porosity into account which is calculated based on smoothness.
In case you have enabled “Rain Details” in the material rain amount also influences the size and number of rendered rain drops which appear on the surface.

Please note: Although it might look quite handy to use this feature to adjust the reflectivity and boost the specular highlights of your plants i would always recommend to adjust their textures instead as rain does not come for free.

Lighting Settings

As the foliage shader supports pretty sharp specular reflections (both direct and indirect) specular highlights might cause very heavy flickering on small or high frequency details like e.g. the nettles provided with the package. In order to get around this the shader fades out **Smoothness** according to the distance to the camera.

As well as specular highlights **Translucency** might look odd at far viewing distances – especially beyond the shadow distance. So the shaders fade out translucency as well.

Range Distance to the camera at which smoothness and translucency will be set to 0.0 on any mesh using the foliage shader.

Fade Length Distance over which smoothness and translucency will be faded out.

Please note: Due to Unity's built in BRDF even a smoothness of 0.0 will give you some specular reflections. That is just how pbs works.

Layer Culling Settings

Layer Culling will help you to enormously optimize performance:
"Normally Camera skips rendering of objects that are further away than farClipPlane. You can set up some Layers to use smaller culling distances using layerCullDistances. This is very useful to cull small objects early on, if you put them into appropriate layers."
http://docs.unity3d.com/Documentation/ScriptReference/Camera-layerCullDistances.html

Enable Layer Culling

If checked Camera Layer Culling is enabled. Changes will only be performed "On Start".

Small Detail Distance Distance at which all game objects placed on the layer defined by "Small Details Layer" will be culled. Changes will only be performed "On Start".
Make sure all objects that should be culled at this distance are assigned to the correct layer.

Small Details Layer Layer you want to be culled at "Small Detail Distance".

Medium Detail Distance Distance at which all game objects placed on the layer defined by "Medium Details Layer" will be culled. Changes will only be performed "On Start".
Make sure all objects that should be culled at this distance are are assigned to the correct layer.

Medium Details Layer Layer you want to be culled at "Medium Detail Distance".

Please note: Using Layer Culling is especially important on manually placed grass as it will fade out according to the detail distance but unlike grass placed using the terrain engine it will not be culled or skipped! Even if it is not visible any more all vertices will be sent to the GPU and processed unless you put all grass objects into a special layer and enable layer culling.

Special Render Settings

The script will configure the "AfsGrassShader" shader used on manually placed grass objects:
- In edit mode it tells the shader to use the model's rotation to simulate ground lighting.
- In play mode, when ground normals have been baked into the meshes, the script makes the shader use those baked normals to lit the models.

All Grass Objects Combined If you have combined all grass objects statically you can tick this check box so lighting will be correct even in the editor.

CombineChildrenAFS Script

This script lets you combine several meshes and game objects into one mesh/one game object (aka patch) which in most cases dramatically increases rendering performance. It also

bakes the needed ground normal for objects using the grass shader into the combined mesh and adds some color variation to objects using the foliage shader.

In order to use this script create a new and empty game object in the hierarchy. Then parent all game objects you want to be batched into one patch under this game object and attach the script.

The combine process will either take place on start (lower scene/built memory usage, higher latency) or in the editor (higher built memory usage, no latency). In case you want to use light mapping or GI you have to statically combine the meshes in the editor.

If the script detects several materials it will create several child objects: one per material. So i highly recommend to use texture atlases and shared materials.

Important: If you want to place grass on top of arbitrary geometry that geometry must have a collider attached. Otherwise the script will not be able to calculate the ground normal.

Important: If you want to use the CombineChildren script none of the child objects must be marked as “static”.

To combine or not to combine

Even if Unity supports dynamic batching and–since version 5.4.–GPU instancing rendering hundreds or even thousands of small instantiated game objects like manually placed grass will most likely kill your framerate and create a vast amount of overhead.

So there is no faster way to render these but combining them into several patches.

Memory Usage

However combining instances increases memory usage as–instead of drawing several clones referencing the same mesh data–each cluster uses a unique mesh. So using a vast number of combined meshes in your scene will most likely eat up your memory.

The current solution to get around the memory issue is to use streaming:

Combine the meshes and objects statically. Then move them to different scenes which get loaded and unloaded depending on if they are needed or not.

Best Practise

- Complex models such as the banana which are spread sparsely across your scene should most likely be placed as single game objects. This will keep memory consumption low and let you take full advantage of frustum based culling and GPU instancing.
  Tip: *You might consider using 3rd party pooling solutions to optimize performance.*
- Simple and high frequent models should be combined into several patches of e.g. 15x15 meters in size.
- Manually placed grass always has to be combined.

	Foliage as GO	**Combined Foliage**	**Foliage on Terrain**
Instancing	yes	no, not needed	no, not needed

Touch Bending	yes	no	no
Casts Shadows	yes	yes	no
Color Variation	yes	yes	yes
Lightmaps/GI	yes	yes	yes – one sample per instance
Preserve Length	yes	yes (!)	no

Patches

When talking about "several patches" it is important that you keep in mind that the fastest drawn mesh is the mesh which isn't drawn at all but simply culled.

The larger the size of your patches the higher the chance that a patch will be send to the GPU although just a view triangles of it are within the given camera frustum. This gets even worse in case you use forward rendering and multiple lights as the patch would have to be rendered multiple times.

So you have to find the right balance between patch size and the number of draw calls: If you have a very dense vegetation it might be better to have smaller patches so none visible patches will be completely culled whereas when you have only a few objects per cluster the wasted calculation on invisible triangles might cost less than invoking another draw call.

Script Properties

Ground normal sampling

Max Ground Distance Max distance in which the script looks for any collider to sample the ground normal from.

Underlaying Terrain If you place the objects of the patch (all or just a few of them) on top of a terrain you will have to assign the according terrain in order to make lighting fit 100% the terrain lighting.

Doing so will force the script to sample the ground normal not using the normal detected by ray casting whenever the terrain is hit. It will use the normal sampled from the given terrain using the "GetInterpolatedNormal" function instead. Only this will make objects using the grass shader blend seamlessly with grass placed within the terrain engine.

In case all objects of the patch are placed on top of arbitrary geometry you do not have to assign any terrain.

Grass Shader Settings

Healthy Color If you use the advanced grass shader you can bake healthy and dry colors to the meshes just like you know from the terrain engine.

Dry Color If you use the advanced grass shader you can bake healthy and dry colors to the meshes just like you know from the terrain engine.

Noise Spread This controls the blending between healthy and dry color. Larger values will give you more variety on smaller clusters.

Foliage Shader Settings

Random Pulse Max. amount of random pulse value that will be added to each child object. Lets you create a bit more variety as far as the secondary bending is concerned.

In case you use UV4 it will also add variation to primary bending per child object.

Random Bending Max. amount of random bending value that will be added to each child object. Think of it as if it was some kind of stiffness: So random bending set to 1.0 might cause some plants to not bend at all.

Random Fluttering Max. amount of random fluttering value that will be added to each child object.

Random Occlusion Max. amount of random ambient occlusion that will be added to each child object. Lets you create a bit more variety in shading.

Noise Spread This controls the variation of all random values. Larger values will give you more variety on smaller clusters.

Bake Scale If scale is baked to the combined mesh larger instances will get more bending whereas smaller instances will get less.

Overall Settings

Cast Shadows Shall the combined mesh cast shadows?

Destroy Children Shall Child objects be destroyed? Usually yes as they do not have any colliders attached. All that we need is the mesh!
Important: If checked all children will be destroyed when you hit "Combine statically". You wont be able to edit them any more.

Use Lightprobes Check this if you want to use lightprobes on the combined mesh.

Debugging

Debug sampled Ground Normals Check this if you would like to see which normal will be baked to which object.

Red lines will show the sampled normal by ray casting, green lines will show the range in which the scripts searches for a collider.

If you have assigned a terrain to the script and the mesh collides with this terrain you will see blue colored normals next to the red ones showing the normals sampled from the terrain using the "GetInterpolatedNormal" function.

If no collider is found within the given range a standard vector.up is displayed (and applied). This will be colored in yellow.

Functions

Realign Ground Max Distance Max distance in which the script looks for any collider in case you use the "Realign Objects" function.

Force Realignment If checked the script will search for the nearest collider looking from above the current object's position when using the "Realign Objects" function even if it is already aligned perfectly.

Use this in case you want to make your objects being aligned to a collider which is above the collider they are currently attached to.

If you want to make your objects being attached to a collider below the current one simply deactivate the upper collider when using the "Realign Objects" script.

Realignment Objects Details >

Combine statically Details >

Create unique UV2 (needed by lightmapper) Check this if you want to bake lightmaps for your plants. UV2 will only be generated if you hit “combine statically”.

Has been statically combined Will be checked by the script automatically if you combine the children statically. The script will still be present but not do anything on startup if checked.

Save Lets you save the created combined mesh to disk so you can easily assign it to e.g. asset bundles.

Please note: You have to combine the children statically before you can save the mesh to disc. Otherwise the script will throw an error.

Realign objects

If you apply any changes to your terrain or the underlying geometry you can automatically realign all objects according to those changes.
In order to do so choose "Realign Objects" from the "gear-wheel" dropdown in the upper right corner of the script or use the button at the bottom of the custom editor.
Please note: You will probably have several colliders layered on top of each other in your scene – like a terrain and some geometry above.
This might lead to problems when realigning the objects: They could change from being aligned to the first to being aligned to the second collider if the "Realign Ground Max Distance" is too big. Adjust this parameters or simply deactivate those colliders which you do not want to check against when using the "Realign Objects" script.

Combine statically

When done with your work on the scene you can combine all submeshes statically so they will not be combined at start anymore. In order to do so choose "combine now" from the "gear-wheel" dropdown in the upper right corner of the script or use the button at the bottom of the custom editor.

Mark the game object as "static" afterwards in order to enable static batching.

Important: If "Destroy Children“ is checked all child objects will also be destroyed when choosing "combine now".

Important: You will not see baked vertex colors (like dry and healthy colors) unless you hit "play" or mark "All Grass objects combined" in the setup script, as the shader does not take those into account in edit mode.

Important: In order to make GI work on the combined mesh you have to combine it statically.

Make sure you have chosen an appropriate layer in order to take advantage of layer based culling.

If you have statically combined all objects using the advanced grass shader you may check "All Grass Objects Combined" in the "Setup Afs" script, This will make the grass shader always render as it does in play mode.

Important: Plants which have touch bending attached may NOT be combined.

FAQ

Lighting of grass seems to be wrong. It is too bright or too dark.
What can I do?

Reason for this will probably be a "wrong" sampled ground normal. Depending on your scene and settings the "CombineChildrenAFS" Script might not find what you consider the right ground normal but a normal from any collider above or below your ground.

The script assumes that your grass prefabs intersect with the ground and that their pivots are just a little bit below or above the ground and it looks for any collider within the "Ground Max Distance".

So start checking the sampled normals by activating "Debug Normals" in the script settings.
If the normals do not look correct you might:
- reposition your grass objects
- adjust the "Ground Max Distance" parameter of the script.
Make sure that your "ground" has a collider attached.

Some of my grass or plant objects disappear in play mode. What is wrong?

If you use the "CombineChildrenAFS" Script make sure that all child objects and the parent are NOT marked as “static”.

There also might be animation components attached to the prefabs. If so please remove them.

Whenever i hit "play" grass gets black. What is wrong?

Make sure that the "CombineChildrenAFS" Script it attached and enabled on the parent object of your grass instances as whenever you enter the playmode the shader changes lighting and assumes that the ground normals are baked into the regular mesh – which is done by the "CombineChildrenAFS" Script.

I have combined grass statically but i do not see any dry and healthy colors. What is wrong?

You will not see baked vertex colors (like dry and healthy colors) unless you hit "play" or mark "All Grass objects combined" in the setup script, as the shader does not take those into account in edit mode.

I get errors in the console. What is wrong?

If the console throws an error like this:
`Not allowed to call GetTriangles() on mesh 'Combined Mesh (root: scene)'`

you have most likely marked one of the objects that should get combine as “static”.
Please make sure that none of the children is marked as “static”.

Model and Texture Guide

In order to be able to use your own vegetation models with AFS they have to fit certain requirements.

- In order to save mesh memory and speed up rendering you should consider to always use **single sided geometry** as the shaders support proper lighting even for that. Only in case you need very fine control over ambient occlusion baked to vertex color alpha or you want to use lightmap based GI you will need double sided geometry.
- **Vertex colors** have to be set up properly to support **bending**.
- If you want to benefit from **batching** plants you want to place close to each other should share the same **texture atlas**.

Geometry

As mentioned above i recommend to always use single sided geometry in order to safe mesh memory and lift some work from the vertex shader.

Make sure that your normals are set properly and define smoothing groups when needed.

Bending

The foliage shaders support complex bending as you might already know from the built in tree creator shaders. It is controlled by the vertex colors applied to the model and/or UV4 coordinates and consists of 3 blended animations:

1. Primary or main bending, which animates the entire model along the wind direction.
2. Secondary or detail bending, adding a higher frequent animation mostly along the y-axis.
3. Edge fluttering which is a “super” high frequent bending animation originally invented by Crytek to simulate single leaves on large leave planes. As we are mostly talking about small to medium sized foliage here where you do not have that large planes , you may simply forget about this… But in case you have rather simple geometry this feature is still at your hand.
4. In order to make the whole blending more believable, there is a fourth factor: per leaf / per branch phase variation.

Bending Modes

The advanced foliage shader support three different bending modes:

- **Vertex Colors** All bending information is stored in vertex color rgba. Baked ambient occlusion is not supported. This will give you less control over the overall look but a pretty small number of vertex attributes.
  Please note: Bending driven by vertex colors supports plants being placed using the terrain engine.
- **UV4 and Vertex Colors** Primary and Secondary bending are stored in UV4, edge fluttering and per branch phase variation in vertex color green and red while alpha stores ambient occlusion. This gives us full control over bending and allows us to even

bake ambient occlusion but does not work for plants you want to place using the built in terrain engine. It also adds slightly more data per vertex.

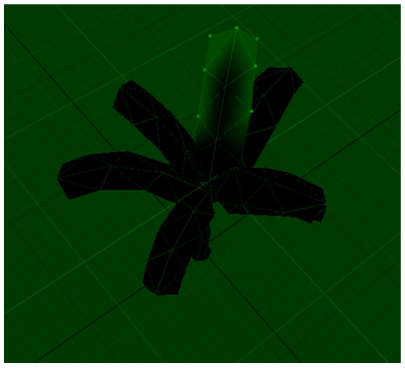
- **Vertex Colors legacy** Here all bending information is stored in vertex color rgb plus ambient occlusion in alpha. As we only have 3 colors for 4 bending parameters we have to combine main and detail bending in vertex color blue. This will give us less control over the overall look but a pretty small number of vertex attributes.
  Please note: Legacy bending supports plants being placed using the terrain engine. Legacy bending will be skipped in the next updates.

	Vertex Colors	UV4 and Vertex Colors	Vertex Colors Legacy
Usage	works on manually placed foliage and foliage placed within the terrain engine	works on manually placed foliage only	works on manually placed foliage and foliage placed within the terrain engine
Memory	small vertex data footprint	slightly higher vertex data footprint (+8 bytes)	small vertex data footprint
Bending Control	full control over primary and secondary bending	full control over primary and secondary bending	no separate control over primary and secondary bending
Baked AO	not supported	supported	supported
Touch Bending	supported	supported	supported
Bending Variety		more variety between adjacent plants if combined	

General Bending Parameters

No matter if you have chosen to go with legacy or new bending—you will have to set up “Edge fluttering” and “per-leaf / per-branch phase variation” manually.

Setting up Edge Fluttering

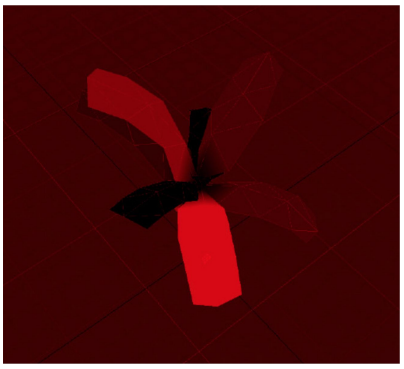


Edge fluttering means just deforming the edges, using the vertex color’s **green** channel for controlling edge stiffness: green=0 --> no fluttering / green=1 --> full fluttering.

Add edge fluttering to leaves by simply adding some green to the outer vertices of the leaves.

Tip: *Make sure that the connecting point (or pivots) of the leaves have vertex color green=0. Otherwise leaves might loose connection to their parent geometry.*

Setting up Phase Variation



In order to avoid that all leaves or branches of your model bend to the exact same pulse, you should add different shades of **red** to single leaves.

Tip: *When using legacy bending make sure that the connecting points (or pivots) of the leaves have vertex color red fitting the vertex color red of the point they are connected to. Otherwise leaves might loose connection to their parent geometry.*

Adding Ambient Occlusion

Ambient occlusion might be baked into vertex color alpha – depending on the bending mode you want to use. How you will do this is up to you and depends on your 3d App. In order to bake ao within unity you can use a small script, which can be found it here: http://adrianswall.com/shared/unity_vertex_ao.rar

Adding primary and secondary Bending

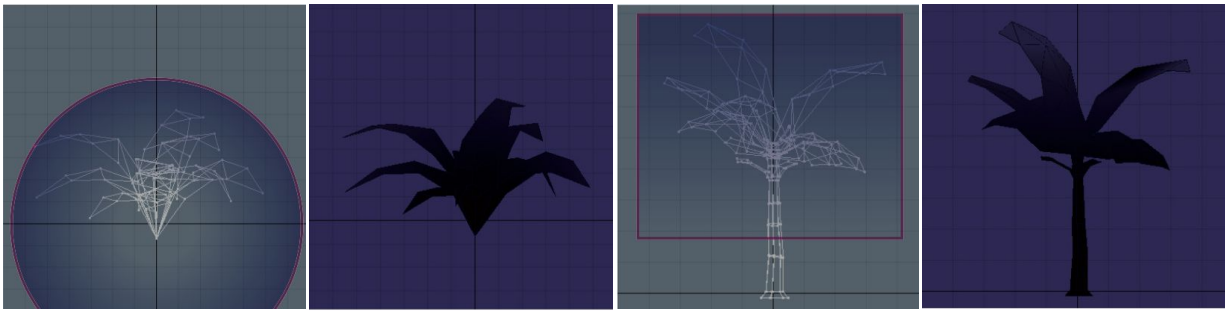
When setting up main and detail bending you should keep in mind that wind settings like direction and strength for all different kinds of foliage are equal – but different kinds of plants might react differently to wind according to their overall size, size of leaves, stiffness and so on and so forth.

For this reason all plants within your scene need their own main and detail bending which means: All different plants do have to have unique maximum bending values in order to achieve a variety in bending. So please constantly compare all models and their bending to each other within Unity during the process of adjusting the vertex colors.

Using vertex colors or vertex colors legacy

The way how to apply vertex color blue and vertex color alpha depends on what kind of plant you are working on.

The package ships with two different models marking the two poles of supported geometry: Whereas the fern model is the most simple one, the small banana tree is a rather complex one, for it consists of a rather stiff trunk and pretty flexible leaves.



Setting up the fern model (legacy bending)

As we do not want to have any bending at its pivot but a lot of it at its outer leaves we simply assign a radial gradient from blue=0 at the pivot to e.g. blue = 0.5 at the tips.

Setting up the banana tree (legacy bending)

Simply assign a linear gradient from blue=0 at one third from its pivot to e.g. blue = 0.7 at its upper parts.

Tip: *When you start adding vertex colors for main and detail bending to your first model do not assign the highest possible value (which would be blue = 1.0) but start right in the middle: max blue = 0.5. That leaves you some space for plants with less or more bending.*

Using UV4 and vertex colors

In case you decide to use UV4 and vertex colors to control bending all vertex colors you have to setup up front are red and green to control edge fluttering and phase variation.
Setting up UV4 in any 3d App would be pretty cumbersome but can be done within Unity using the AFS Foliage Tool. However you may apply vertex color blue to mask secondary bending on certain vertices: Set vertices you want to exclude to vertex color blue = 0.0 or use a value < 1.0 to weaken secondary bending.

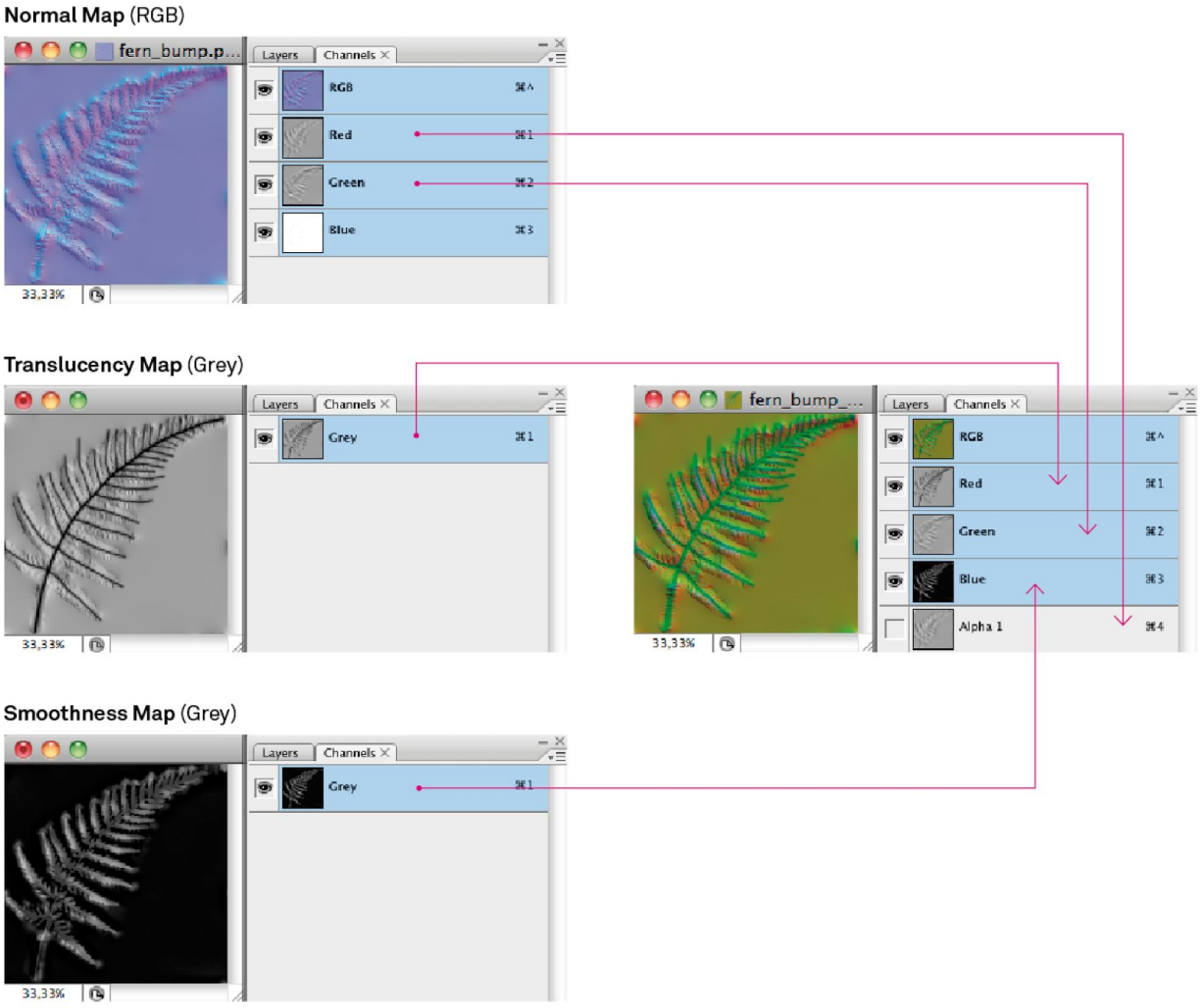
Using the AFS Foliage Tool to adjust primary and secondary bending

You may just set up edge fluttering and phase variation using your 3d app—and use the Foliage Tool to set up main and detail bending right within Unity. The foliage tool also lets you adjust already baked vertex colors for edge fluttering, phase variation and ambient occlusion. Please have a look at the corresponding chapter to find out more.

Textures

Creating combined textures for the foliage shader

The foliage shader supports diffuse, alpha, normal, smoothness and translucency maps. But in order to keep the texture load and bandwidth footprint as small as possible those are combined into 2 textures: Diffuse/Alpha and Normal/Translucency/Smoothness.
You will have to generate the combined Normal/Translucency/Smoothness texture manually like shown in the figure below.



Combined Normal/Translucency/Smoothness Texture Color Channel Layout

Please note: These combined textures have to be imported using *Advanced -> Bypass sRGB Sampling* checked.
As they contain normal maps you should also set the *Filter Mode = Trilinear*.

Creating combined textures for the grass shader

As grass does not support normal mapping nor direct specular lighting we do not need normals nor smoothness. But you may define translucency and ambient occlusion for grass. **Translucency** goes into the **green** color channel, **occlusion** into the **blue** color channel of the combined texture.

Texture Atlasing

In order to let the “Combine Children AFS” script combine different plants into one mesh those plants have to share the same material and textures. So you will have to create a texture atlas for those plants and map all models to that atlas.

Please note: In case you want to use the foliage shader within the terrain engine or just have different grass textures but want to take fully advantage of features such as translucent lighting on grass creating a texture atlas for all plants placed within the terrain engine is mandatory.

Grass and Foliage shaders for the terrain engine

Usually you would just add single textures or models to the terrain and let the terrain engine create a texture atlas for all given details automatically. Unfortunately this will not work in case you want to use the grass and foliage shaders within the terrain engine because they need a second texture containing the combined Normal/ Translucency/ Smoothness maps.

For this reason you will have to create the texture atlas manually and map all objects—no matter if they use the grass shader or the foliage shader—to that atlas.

You also have to author a second atlas containing all combined textures.

That being said you might already got it: In case you want to take full advantage of the grass shading features or even want to add some plants using the foliage shader to the terrain you may NOT add any grass as simple texture–but **always have to add it as mesh** using the combined texture atlas. Billboarding grass is not supported.

Please note: Placing meshes within the terrain engine implies all disadvantages placing meshes within the terrain engine usually does: They do not cast shadows and they do not have any collider.

	Manually added foliage	**Foliage in the terrain engine**
Bending	can be controlled by vertex colors and/or UV4	must always be controlled by vertex colors (legacy)
Casts Shadows	yes	no
Bump maps	fully supported	fully supported
Touch Bending	yes	no

Reading all this restrictions, limits and requirements you might ask: Why should I use any of the features within the terrain engine at all? The answer is simple: Grass will just look so much more beautiful using translucent lighting and proper translucency textures! So it is definitely worth taking all the obstacles :-)

When it comes to foliage it is a bit more difficult as foliage placed using the terrain engine not casting any shadows at all is a real downer. Nevertheless this feature lets you place vast

amounts of e.g. simple plants like flowers or clover which will just beautifully sway in the wind and do not necessarily cast shadows.

Please note: Even if you are using the “hacked” vertex lit shader on foliage placed on the terrain you may also add objects which shall not react to wind like small rocks and stones. All you have to make sure is that their models have proper vertex colors baked into them which would be: rgba = 0,0,0,0.
You may also just set “Healthy” and “Dry Color” to pure black.
Please have a look at the “Foliage Shader Overview Demoscene” and the little rocks added to the terrain to find out more.

In addition to that **“Healthy”** and **“Dry Color”** on meshes using the hacked vertexLit Shader will not apply any color on the instances but influence bending!
Red changes the bending phase, **Green** changes the edge flutter, **Blue** changes main and secondary bending in case you use legacy bending. In case you use bending mode = vertex colors **Blue** will influence secondary bending while **Alpha** adds changes to the primary bending.

Foliage Tool

The Foliage Tool lets you adjust the baked main bending parameters of your plant meshes or even set them up “from scratch” right within Unity. It also lets you convert trees created using the built in tree creator into simple trees or to be compatible with the foliage shader.

Setting up a test scene

In order to adjust bending parameters within Unity i recommend to create a little test scene which should contain almost all of your plant models as it is very important to judge the bending of a certain plant relatively to all others.

Of course you have to add the "AFS Setup Prefab” too. Otherwise you won’t see any bending at all. Set up the wind so that it will fit your general settings.

Using the script to adjust foliage assets

Select the plant/prefab you would like to edit, drag it to the scene view and choose "Component" --> “AFS“ —> “Mesh" --> "AFS Foliage Tool".

A custom editor will become visible in the inspector which will ask you to save a new mesh asset first. When done you will get two foldouts giving you two different ways to edit the new mesh:

Please note: Bending is driven by the baked bending parameters, the material set up and your current wind settings. So please keep an eye on all these inputs.

Tip: *check “Animated Materials” in the View tab. This will give you a pretty good preview of how bending will look like in play mode. However when hitting test and entering play mode you will also see how the materials Leaf Bending parameter influences the overall behaviour.*
Do not forget to rotate the model or wind and check it from all sides!

Header

Original Mesh Should automatically be filled with a reference to the original mesh, but you may assign one manually too.

Revert entire Mesh In case there is a reference to the “Original Mesh” pressing this button will revert and/or update the working mesh. Do so in case you do any changes to the original mesh like adding or deleting leaves etc.

Please note: Vertex colors will be restored. Bending information you added to UV4 will be cleared. But as your settings are still present you may simply hit “Apply” to update the reverted or updated mesh. This of course does not work if you have edited the vertex colors in several steps...

Show Vertex Colors If checked the currently assigned material will be set to use the debug shader which outputs the unlit vertex colors.

Bending Mode Lets you choose where primary and secondary bending should be stored:

- **Vertex Colors** Primary and secondary bending will be stored in vertex color alpha and blue so you can adjust them separately.
  Please note: Meshes which should be added using the terrain engine have to either use this method or Vertex Colors legacy.

- **Vertex Colors And UV4** Primary and secondary bending will be stored in UV4 so you can adjust them separately. This gives you better control but will increase the amount of data per vertex.
- **Vertex Colors legacy** Primary and secondary bending will both be stored in vertex color blue which gives you less control over bending but a smaller vertex data count.
  Please note: Meshes which should be added using the terrain engine have to either use this method or Vertex Colors.

Set bending

Open this foldout if you want to completely redefine primary and secondary bending.

Blueprint In case you want to create LODs or simply copy settings from another prefab (which must have the Foliage Tool script assigned to) assign the prefab here.

Copy Settings from Blueprint will copy all settings from the blueprint to the script.

Grab Bounds As all calculation is done relatively to the mesh's bounds you may grab the blueprint's bounds and make all calculations relative to those. This is useful in case you want to create LODs and the meshes for the different LODs have slightly different bounds.

Reset Bounds Assign the original bounds.

Depending on the chosen ***Bending Mode*** you will get different inputs:

Bending Parameters = Vertex Colors or Vertex Colors And UV4

Primary Bending

Along Y_Axis Sets up primary bending which will be stored in UV4.x according to the y-position of the given vertex (slider defines max strength, curve defines the strength along the given axis). Suitable for more or less upright orientated plants.

Relative To Pivot Sets up primary bending which will be stored in UV4.x according to the distance between the pivot and the given vertex (slider defines max strength, curve defines the strength along the given axis).

Suitable for "bushy" plants.

Secondary Bending

Along XZ-Axes Use this if you want to add stronger bending to the outer parts of your plant. It will store secondary bending values in UV4.y according to the vertex position along the xz-axes relative to the pivot. (Slider defines max strength, curve defines the strength along the given axis).

Mask secondary Bending

Along Y_Axis Select this and adjust the related curve to prevent vertices to get any secondary bending applied to according to their y-position–or simply lower secondary bending on certain parts.

by Vertex Color Blue Select this if your model has been set up using vertex color blue to mask or define secondary bending.

In case even strength has been baked to vertex color blue you might set the curve which control secondary bending along the xz-axes (green curve) to a flat line by simply applying the

first curve preset in the curve editing window. You still may adjust the overall strength using the corresponding slider.

Bending Parameters = Vertex Colors legacy

Primary and secondary Bending

> **Along Y_Axis** Sets up primary and secondary bending according to the position of the given vertex along the y-axis relative to the pivot (slider defines max strength, curve defines the strength along the given axis). Suitable for more or less upright orientated plants.
>
> **Relative To Pivot** Sets up primary and secondary bending according to the distance between the pivot and the given vertex (slider defines max strength, curve defines the strength along the given axis). Suitable for "bushy" plants.

Along XZ-Axis Use this if you want to add stronger bending to the outer parts of your plant. It will add primary and secondary bending according to the vertex position along the xz-axes relative to the pivot (slider defines max strength, curve defines the strength along the given axis).

Show Gizmos If checked you will see the bounding box of the current mesh next to some colored markers – each representing a key of the curves that shares the same color.

Apply Hit "Apply" to bake the new values to the temporary mesh.

Test Hit "Test" to see how wind will bend the temporary mesh in play mode.

Important: Do not forget to save the modified mesh when you are done!

Adjust vertex colors

Open this foldout if you just want to adjust already existing vertex colors.

Please note: All calculations are done relatively to the currently defined snapshot.

Take Snapshot Defines the current vertex colors as new basis which all further calculations will be done relatively to.

Revert to Original Revert to the vertex colors stored in the mesh assigned as **Original Mesh.**

Most important part of this section are the **RGBA curves** which let you tweak the brightness and contrast of the vertex colors just like in Photoshop.

Apply Hit "Apply" to bake the new values to the temporary mesh.

Test Hit "Test" to see how wind will bend the temporary mesh in play mode.

Important: Do not forget to save the modified mesh when you are done!

Using the script to convert "tree creator" trees

In case you want to use the advanced foliage or simple tree shader on models created with the built in tree creator you might use the foliage tool to make those models fit the needs of those shaders.

Why should i use the advanced foliage shader on trees or convert a tree into a flattened tree?

A lot of packages on the asset store ship with rather small plants modeled using the tree creator like bushes or even small trees. Using the original tree creator shaders will usually

force 2 draw calls per instance (4 when using real time shadows) — 1 for bark + 1 for leaves—whereas when converting the tree to a "flattened Tree" it is only 1 (2) draw call as all materials are flattened and everything is rendered using the leaf shader.

Using the "foliage" shaders in conjunction with the Combine Children AFS script might give you even less draw calls.

So if you do not need absolutely correct lighting (as bark is lit as leaves) converting those models will save you a lot of draw calls.

Converting a tree into a flattened tree

Cons

- Only one shader (leaves) will be used: So “wrapped around diffuse” lighting will be applied even to the bark and make its general lighting and bump mapping “smoother”.
- You may get translucency even on parts of the bark which can easily be fixed by editing the applied textures—or will be fixed by the script automatically.

Pros

- Just a single draw call per tree instead of 2 (or 2 instead of 4 if you have real time shadows enabled)
- Billboarding and wind zones (directional and radial) are fully supported

Recommended Use Case

Use this option on trees which do not not need "correct" and sophisticated lighting on the bark (like bushes where the leaves most likely hide most of the bark anyway) but which should be visible over a long viewing distance (billboarding).

Converting a tree into a mesh compatible with the foliage shader

Cons

- Meshes using the foliage shaders do not react to (radial) wind zones but only to global (directional) wind parameters as defined by the "Setup Afs" script.
- The foliage shaders will give you slightly different lighting on the bark.
- If you select: "vertex colors" this will lead to inferior bending which even has to be set up manually. If you select: "vertex color and uv4" however bending will be more or less the same without any needs to tweak it – perfect!
- No more billboarding as the result will not be a tree anymore. These meshes should be culled early using layer culling.

Pros

- Pretty fast rendering: Just a single draw call per "tree" instead of 2. GPU instancing is fully supported. Alternatively you might use the "Combine Children Afs" script.
- Touch Bending might be enabled.

Recommended Use Case

Use this option on "trees" which do not have to be visible over a long viewing distance and benefit from GPU instancing or use the "Combine Children Afs" script to group those “trees” into patches in order to reduce the number of draw calls even more:

Just imagine a dense forest where you have e.g. 20 bushes in an area of 10 x 10 meters: Instead of 20 draw calls you will just get 1.

__Tip:__ As trees generated by the tree creator usually use double sided geometry on the leaf planes you should set “Culling” to “Back” in the material properties to speed up rendering.

Please note: Using the "Combine Children Afs" script will increase mesh memory usage.

How to convert a tree

Important: First of all you should duplicate the tree’s prefab in the project tab.

Then drag the copy into your scene and apply the foliage tool to it by choosing "Component" --> „AFS“ —> “Mesh" --> "AFS Foliage Tool".

A custom editor will become visible in the inspector.

The script will ask you to save a new mesh asset first. When done the script will detect that your mesh consists of 2 sub meshes (for the 2 materials: leaves and bark) it will prompt a proper message.

You will also get two foldouts giving you two different ways to convert the tree:

Convert Tree into flattened Tree

Mesh Processing Nothing to do here. The script simply flattens the sub meshes and copies all bending parameters.

Texture Processing

> **Texture split** If your tree uses one of the most common texture setups the script can automatically fill the proper parts of the translucency map with black so the bark will be completely opaque. Otherwise you will have to edit the generated texture manually and fill those parts of the red color channel with black where the tree should be fully opaque.
>
> > **One by One** bark on the left half / leafs on the right half
> >
> > **Two by One** bark fills 2/3 of the texture atlas / leafs the third on the right side of the texture atlas
> >
> > **Other** If selected the script will not make any changes to the translucency channel (red). You will have to edit it manually and fill those parts where there shall not be any translucency with black.
> >
> > **Please note:** If the original bark texture contains any alpha channel which is not purely white (which would be fully opaque) the bark will not show up correctly as it will be rendered using alpha cutoff.
> >
> > Fix this by duplicating the original "diffuse" texture created by the tree creator and editing its alpha channel: Please set all parts where there is the bark texture to pure "white".

Convert Tree Hitting this button will start the process.

When finished simply remove the script and create a new prefab.

Please note: You have to keep the “Tree” component attached as otherwise the tree would not be affected by wind.

Convert Tree for Foliage Shader

Mesh Processing

Bending Parameters Lets you choose between "Vertex Colors" and "Vertex Color And UV4":

> **Vertex Colors:** Primary and secondary bending will be stored in vertex color alpha and blue. Baked ambient occlusion gets dropped.

Please note: Meshes which should be added using the terrain engine all have to use either this method or **Vertex Colors legacy.**

Vertex Colors And UV4: Primary and secondary bending will be stored in UV4. This will slightly increase the amount of data per vertex but all bending values are straight copied so you do not have to do anything after conversion.

Vertex Colors legacy: Primary and secondary bending will both be stored in vertex color blue which gives you less control over bending but slightly less data per vertex.

You will have to set up primary and secondary bending manually afterwards.

Please note: Meshes which should be added using the terrain engine all have to use this method or **Vertex Colors.**

Please note: If you convert a tree and bake bending to vertex colors only you will loose some baked bending information and have to set up primary and secondary bending from scratch. However as trees usually contain very different values of vertex color red in their different parts like trunk and branches your tree will most likely break apart: Lower the vertex color red values to get around this. This will give you less variety but a stable tree.

Generate UV2 If checked UV2 will be generated (needed by light mapper). In case you will use the "Combine Children Afs" script this lets you create UV2 too.

Texture Processing

Texture split If your tree uses one of the most common texture setups the script can automatically fill the proper parts of the translucency map with black so the bark will be completely opaque. Otherwise you will have to edit the generated texture manually and fill those parts of the red color channel with black where the tree should be fully opaque.

One by One bark on the left half / leafs on the right half

Two by One bark fills 2/3 of the texture atlas / leafs the third on the right side of the texture atlas

Other If selected the script will not make any changes to the translucency channel (red). You will have to edit it manually and fill those parts where there shall not be any translucency with black.

Please note: If the original bark texture contains any alpha channel which is not purely white (which would be fully opaque) the bark will not show up correctly as it will be rendered using alpha cutoff.

Fix this by duplicating the original "diffuse" texture created by the tree creator and editing its alpha channel: Please set all parts where the is the bark texture to pure "white".

Convert Tree Hitting this button will start the process.

When finished simply remove the script and create a new prefab or set up primary and secondary bending.

Grass Model Postprocessor

As some 3D applications do not let you set up vertex color alpha properly (e.g. Blender) as needed by grass meshes and the advanced grass shaders this little script will add vertex colors automatically on import and whenever the grass meshes are updated.

In order to let the script find all models that should be processed you will have to name them properly:

The script finds them by name which must contain: „_AfsGM“.

Maximum Bending

You also have to specify the max bending value by adding it directly after „_AfsGM“. Bending values must be in the range of 0.0 – 1.0 and have to be written as 2 digits without the period:

“00“ -> 0.0 / “05“ -> 0.5 / -> “10“ -> 1.0

So a file named:
“MyTallGrass_AfsGM05_test.fbx” would be processed using a max alpha value of 0.5.

A file named:
“MySuperfern_AfsGM03Testnew.fbx” would be processed using a max alpha value of 0.3.

A file named:
“MySuperfern_AfsGM1Testnew.fbx” would throw an error.

Power

In case you have a more complex model which consists of several triangles you may also specify a **power factor**. Adding a power factor will change the linear gradient which gets applied to the alpha towards an exponential gradient and results in much livelier bending.

Do so by adding “POW” + 2 digits to the file name e.g.:
“weed_atlas_AfsGM02POW16.abx”

Processing Details

The postprocessor will add vertex colors to the mesh (in case it does not contain any) and set vertex color RGB to 1,1,1.

Vertex color alpha will be set according to the “local” y position of the given vertex, the max bending and the power value: Vertex color alpha = 0 for all vertices if vertex.pos.y < 0 and a lerped value for all other vertices according their y coordinate divided by the height of the bounding volume.

If you are using those meshes within the terrain engine you will have to “refresh” your details after any update: "Terrain Inspector" —> "Paint Details" —> "Refresh".

Please note: The postprocessor expects a single flattened mesh. It does not merge sub meshes and skips processing on models which have more than 1 sub mesh.

Touch Bending

Touch bending allows objects using the foliage shader to interact with your player or any enemy in your scene. Touch bending is only supported on manually placed game objects and does not work on plants placed using the built in terrain engine or combined meshes.

Setting it all up

Setting up the player and enemies

- You have to attach the **"touchBendingPlayerListener"** script to each game object that shall interact with the plants and set it up:
  "Component" --> “AFS“ —> “Touch Bending" --> "Player Listener".

  Max speed Make sure that its value always is higher than the maximum speed the related game object will ever reach.

  Player_Damp Speed The amount of touch bending is connected to the player’s speed as long as the player is inside the trigger in order to freeze touch bending when the player suddenly stops.
  The Player_Damp Speed parameter lets you control the damping of the player’s speed passed to the touch bending animation in order to get at least some swing back animation although the player has suddenly stopped.
- Make sure that all game objects also have a collider AND a rigid body attached.
- Mark the rigid body as "is Kinematic" in case you do not need it for any other purpose.

Setting up the plant prefabs

- Enabled “Touch Bending” for the material used by your plant’s prefab.
- Attach a simple collider (e.g. a sphere collider) which is marked as "isTrigger" and scale it properly.
- Add the **“touchBendingCollision“** script to your prefab and set it up:
  "Component" --> “AFS“ —> “Touch Bending" --> "Collision".

  Bendability Low values will give you only little bending whereas high values will raise the amplitude.

  Disturbance Amount of extra disturbance (like secondary wind bending) added to the plant while it is touched.

  Duration Duration of the touch bending animation in seconds.

Tip: *Find out more by having a look into the Touch Bending Demo.*

Repositioned terrain trees

If you add large rocks or cliffs to your scene you usually won’t be able to place trees on top of those—at least not trees placed within the terrain engine which benefit from billboarding.

But instead of adding manually placed trees which do not get billboarded and might produce a lot of overhead as each tree would be a single game object you can use the "Reposition trees AFS" script to make your trees added to the terrain appear on top of your manually added geometry.

Setting up the script

Add the "Reposition trees AFS" script to your terrain and set it up like described below:

Additional Colliders Add all objects which you want to place terrain trees on top to this list by dragging them onto the slot "Add new collider". Then hit "Add".

Please note: All game objects added must have a collider attached to (best is a mesh collider). Otherwise the script can not calculate where to place the terrain trees.

Exclude Tree Prototypes from Repositioning In case you have overlapping geometry like an overhang you might want to place some trees on top of the overhang and others below the overhang on top of the terrain. Usually the script would simply place all trees on top of the overhang but you can exclude some tree prefabs from being repositioned by checking their checkbox so they will always stuck to the terrain.

In case you just have one tree you would have to add this prefab twice to the terrain: first one will be repositioned, second one not in case you check it checkbox.

Reposition Trees Hit to make it happen.

Please note: Whenever you make any changes to the height map of the terrain unity will automatically reposition all terrain trees. You have to hit *Reposition Trees* again to make the tree appear on top of the geometry.

Adding trees on top of your geometry

The script does not ship with its own tree placement tool. Instead it just uses the built it one. So if you want to place a tree on top of any geometry simply select the "Terrain" --> "Place Trees" Tool, select the tree prefab you would like to add and switch to the scene view.

You will see the blue circle which is your top down projected cursor. So navigate to the area you would like to place your trees, make sure that you see the blue circle at exact the position on top of your geometry where you would like to place the tree and click the left mouse button.

This will add a tree at the given position – but the tree will be placed on top of the terrain and not the geometry. All you have to do is to hit "Reposition trees" to make the tree(s) being placed on top of your geometry.

Geometry Brush

The original version of this script was provided by Matt McDonald, Owner, Chief Creative Officer and President of Heavy Water, www.heavyh2o.com at:

http://forum.unity3d.com/threads/183224-Using-Tree-gameobjects-vs-Unity-s-built-in-system/page10

Thanks a lot for this really nice tool!

I only made several changes mainly in order to make it work with Unity 3.5—5.0.

So this version of the script does not need game objects which have colliders attached—at least not if you just want to paint and delete those. However "Prevent Overlap" and "Spacing" do not work without colliders.

"Random Rotations" always is applied (as setup in the "Geometry Brush Manager" window) even if "Align to Normal" is checked. Simply set all rotation values to "0" if you want to paint perfectly aligned objects.

The modified version allows painting with prefabs which keep link to the original version in the project tab (in the editor only).

Using the Geometry Brush

1. Create a new game object that will hold all painted objects.
2. Add the "GBSettings" Prefab to your scene.
3. Select "GBSettings" in the hierarchy and make it components show up in the inspector.
4. Hit "Open the Geometry Brush Manager"
5. Assign your new game object from step 1. as "Parent Game Object" in the "Geometry Brush Manager" window.
6. In order to add game objects to the brush simply find them in the project tab and mark them.
7. Then come back to the "Geometry Brush Manager" window and click "Add Selections to Brush".
8. Finally adjust your brush settings and start painting.
9. Have fun.

The Shaders

Foliage Shader

As written in the introduction the foliage shader is the most advanced shader of this package—supporting physically based shading and touch bending on manually modeled geometry. New since 5.02: It also support LOD cross fading.

It is suitable for more complex meshes like ferns, bushes or even smaller trees and needs complex bending information to be baked into the mesh.

In order to be able to use the shader with your custom meshes latter have to fit its needs described in the Model and Texture Guide chapter. But just to give you a brief overview: You have to add vertex colors (and may be UV4 coordinates), which control bending. Add those manually or use the Foliage Tool to do so.

Tip: *When using LOD Fading instancing gets disabled by Unity. So i recommend to not create very dense clusters: The number of draw calls might suddenly "explode" when approaching such a cluster.*

Shader inputs

Culling Use "Off" in case you use single sided geometry (recommended). "Back" would be the correct choice for double sided geometry. "Front" is available just because it would be the third possibility…

Color Color multiplier for Albedo and Alpha

Color Variation Lets you you define a color which gets multiplied on top of the albedo and varies from instance to instance. Alpha defines the maximum strength.

Albedo (RGB) Alpha (A) Main diffuse texture (RGB) Transparency (A)

Alpha cutoff If the alpha channel of the Base texture contains different shades of gray instead of just black and white, you can manually determine the cutoff point by adjusting the slider.

Normal (GA) Trans(R) Smoothness(B) This slot holds the combined texture containing the normal map, the translucency map and the smoothness map. You have to author this texture according to the Model and Texture Guide.
Please note: This texture has to be imported using *Advanced -> Bypass sRGB Sampling* checked. You most likely also want it to be sampled using *Filter Mode = Trilinear* as it contains a normal map.

Specular Reflectivity lets you define the specular color (one color for the whole mesh). As plants are most likely common dielectrics values around RGB = 52, 52, 52 should be fine.

Translucency acts as factor which gets multiplied with the translucency value sampled from the “Normal (GA) Trans(R) Smoothness(B)” map and lets you fine adjust final translucency.

View Dependency determines when the translucent lighting effect will kick in depending on the view angle: Lower values will make translucent lighting appear already at rather flat viewing angles while high values will make it appear only if you look directly towards the sun. Values between 0.7 – 0.8 should be fine in case you want some kind of traditional thin layer translucency.

Backface Smoothness lets you adjust the smoothness on back faces. This feature only works with single sided leaf geometry.

Leaf Turbulence Increasing Leaf Turbulence will make those parts of the plant which are affected by secondary bending sway faster if the wind blows stronger.

Bending Parameters Chose between "Vertex Colors" and "Vertex Colors And UV4" according to how the bending information is baked into the given mesh.

Enable Touch Bending In case you want to use touch bending on objects using the given material you have to check this. More about touch bending >

Enable Rain Details Check this if you want raindrops to be rendered due to the given rain amount.

Rain Normal (GA) Mask (B) Assign a proper rain normal and mask texture to this slot if you have enabled **Enable Rain Details.** Assign the provided “Rain Drops Nrm Mask” if you do not have authored your own.

Rain Texture Scale defines the tiling of the **Rain Normal (GA) Mask (B**) texture according to the base UVs.

Baked Pivots Check this in case you have combined several instances using the Combine Children script.

Compatibility

Enable Critias Foliage support Check this in case you place foliage using the Critias Foliage system. By doing so all instances will fade out smoothly towards the given “Max Draw Distance” as specified in the Critias Foliage painter. “Foliage Type” should be set to “OTHER_GRASS”.

Tree Creator Shaders

Assign these shaders to your tree creator trees to replace the built in shaders and benefit from enhanced fading between mesh trees and billboards, physically based shading, physically based wetness [beta] deferred translucency and GPU instancing.

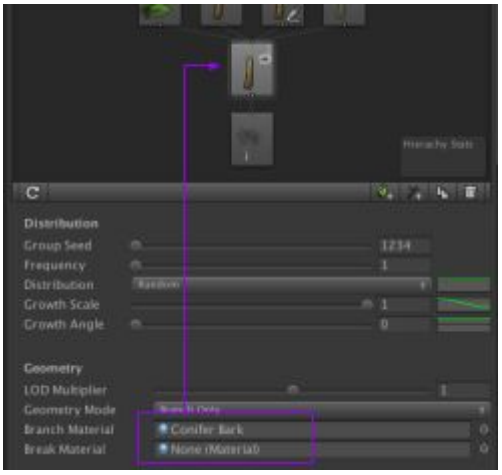
Improved fading

In order to make the fading between billboard and mesh tree as smooth as possible the shaders fade in wind as defined by the *Fade Length* property. During this transition Unity squashes the mesh tree from the original mesh towards a rather flat and crude version which however pretty much fits the tree as it is rendered in the billboard texture.

The squashed version may not cast any shadows as those would look pretty strange. So for this reason shadows as well as translucent lighting and specularity (which both rely on shadows to be present to look correctly) are faded in delayed right after the squash transition has ended.

So the final transition has double the length of the specified *Fade Length* and looks like this:

Assign the shaders



The recommended way is to assign the "AfsTreeCreatorBark" and "AfsTreeCreatorLeaves" shader to the base materials of your trees—so the tree creator will automatically assign the corresponding "optimized" shaders as soon as you hit "update tree".

Base materials are the materials you e.g. assign to the branch nodes (“Branch Material” and “Break Material”) within the tree creator inspector or which are assigned to the leaf meshes you use as prototypes for adding mesh based leaves like shown in the screenshot.

Tip: *Please have a look at the “Conifer [TreeCreator]” tree which ships with the package to find out more.*

Please note: Tweaking any shader input will not affect trees on the terrain unless you refresh the tree prefabs in the terrain engine.

Shader Inputs AfsTreeCreatorBark

Base (RGB) **Occlusion (A)** Assign a proper diffuse texture here. Alpha might contain ambient occlusion though.

Normal Map Assign a regular normal map.

Smoothness (A) Assign a texture which contains smoothness in the alpha channel here.

Shader Inputs AfsTreeCreatorLeaves

Base (RGB) Alpha (A) Assign your diffuse texture here. Alpha must contains a proper alpha mask.

Alpha Cutoff If the alpha channel of the Base texture contains different shades of gray instead of just black and white, you can manually determine the cutoff point by adjusting the slider.

Normal Map Assign a regular normal map.

Smoothness (A) Assign the smoothness map here. Smoothness has to be stored in the alpha channel.

Translucency (A) Assign the translucency map here. Translucency has to be stored in the alpha channel.

Afs specific tree material properties

[Not exposed to the material inspector of the base materials]

As the tree creator only copies fixed coded but no custom values to the optimized tree materials and even locks the optimized materials you have to add the "TreeAfsProperties" script to your tree prefab in order to be able to adjust all Afs related tree shading features.
Tip: *Please have a look at the "Conifer [TreeCreator]" tree which ships with the package to find out more.*

If the script is attached to your prefab you should drag the prefab into the hierarchy as tweaking the properties on the prefab itself surprisingly does not work properly. Now you can tweak the properties on the instance.

In case you want to tweak wetness properties do not forget to raise the global *Rain Amount* property in the Setup Afs Script.

In order to see the changes on trees placed on the terrain you will have to refresh the terrain trees.

Bending

Extra Leaf Bending Values > 0.0 will add some extra bending along the y-axis according to vertex color green (which is also used to define edge fluttering). Your leaf planes should be custom meshes and the pivot point should have vertex color green set to 0.0. Otherwise leaves might get disconnected from the branches or trunk.

Wetness

Dynamic wetness on trees driven by the global *Rain Amount* property in the Setup Afs Script is currently beta. It may be enabled and tweaked per tree prefab and will affect the leaf and bark materials. Wetness is calculated taking the worldnormal up direction, porosity (calculated based on the smoothness) and ambient occlusion into account and will be faded out towards the billboard distance.

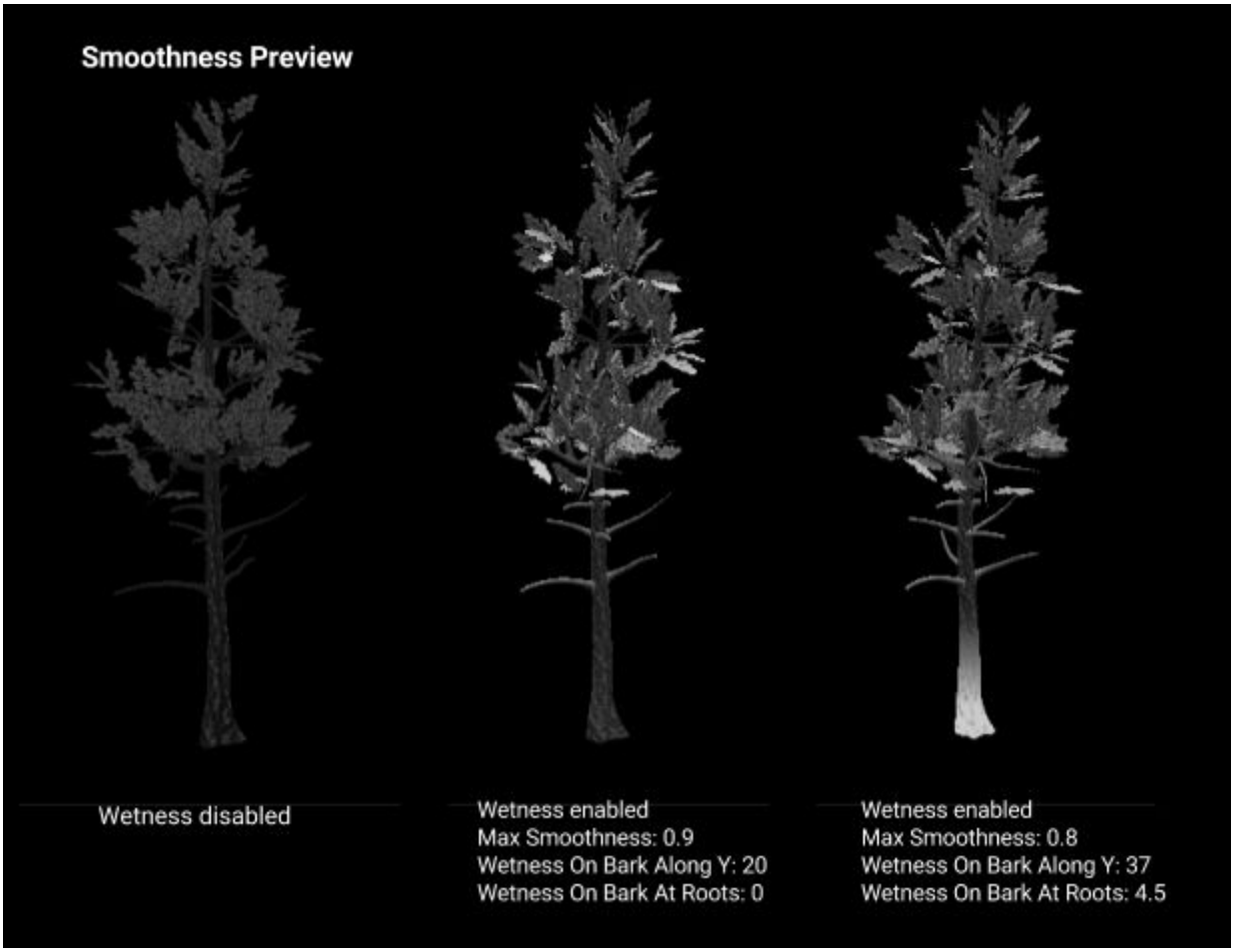
Enable Dynamic Wetness Check this to enable dynamic wetness.

Max Wetness Lets you dampen the accumulated wetness as specified by the global *Rain Amount* property in the Setup Afs Script. This property affects the leaf and bark materials.

Max Smoothness Defines the max. smoothness on fully wet parts of the tree.
If this value is below the base smoothness as defined by texture input raising the *Rain Amount* will actually *dry* the tree… Recommended values are between 0.7 – 0.9. This property affects the leaf and bark materials.

Wetness On Bark Along Y Lets you fade out the wetness on the bark along the y axis of the tree. Vertices above the given value will not get any wetness. This property only affects the bark material.

Wetness On Bark At Roots Lets you add and extend full wetness as defined by *Rain Amount* and *Max Wetness* at the lower parts of the trunk. This property only affects the bark material.



Afs specific material properties of flattened trees

In contrary to *regular* trees flattened trees allow you to edit the final material directly, so you do not need to attach the "TreeAfsProperties" script. Simply by opening the material inspector you will see and be able to tweak the Afs specific material properties.

Enable Dynamic Wetness Check this to enable dynamic wetness.

Wetness Settings X: Max Wetness / Y: Max Smoothness

Max Wetness lets you dampen the accumulated wetness as specified by the global *Rain Amount* property in the Setup Afs Script.

Max Smoothness defines the max. smoothness on fully wet parts of the tree.
If this value is below the base smoothness as defined by texture input raising the *Rain Amount* will actually *dry* the tree… Recommended values are between 0.7 – 0.9.

Please note: As flattened trees do not distinguish between bark and leaf material dynamic wetness on the parts of the flattened tree which are covered by the bark texture will look different compared to regular trees.

Simple Tree Shader

Some of your foliage prepared to work with the advanced foliage shader might be rather big and should be visible over a very long viewing distance—just like our banana.

So in order to reduce rendering costs on plants which are far away in the background but still visible you might either use the built in LOD system or the "AFS Simple Tree" shader.

This shader lets you place models prepared for the usage with the foliage shader (correctly set up vertex colors and/or UV4) as "tree" within the terrain engine so rendering will be speeded up by the internal billboard system of the terrain engine.

Please note: If you just want to simplify existing tree creator trees or convert those to be compatible with the foliage shaders please have a look into the chapter Foliage Tool.

Please note: You can not use this shader on manually placed game objects as they will not billboard nor will they receive any wind. You have to add models using this shader by adding them as tree to the terrain.

Shader Inputs

Color Color multiplier for Albedo and Alpha

Albedo (RGB) Alpha (A) Main diffuse texture (RGB) Transparency (A)

Alpha cutoff If the alpha channel of the Base texture contains different shades of gray instead of just black and white, you can manually determine the cutoff point by adjusting the slider.

Normal (GA) Trans(R) Smoothness(B) This slot holds the combined texture containing the normal map, the translucency map and the smoothness map. You have to author this texture according to the Model and Texture Guide.
Please note: This texture has to be imported using “Advanced” -> “Bypass sRGB Sampling” checked. You most likely also want it to be sampled using “Filter Mode” = “Trilinear” as it contains a normal map.

Specular Reflectivity lets you define the specular color (one color for the whole mesh). As plants are most likely common dielectrics values around RGB = 52, 52, 52 should be fine.

Translucency acts as factor which gets multiplied with the translucency value sampled from the “Normal (GA) Trans(R) Smoothness(B)” map and lets you fine adjust final translucency.

Bending Parameters Chose between "Vertex Colors" and "Vertex Colors And UV4" according to how the bending information is baked into the given mesh.

Extra Leaf Bending Values > 0.0 will add some extra bending along the y-axis according to vertex color green (which is also used to define edge fluttering). Your leaf planes should be custom meshes and the pivot point should have vertex color green set to 0.0. Otherwise leaves might get disconnected from the branches or trunk.

Grass Shader

Use the grass shader on rather small and simple geometry which more or less consists of only some upright oriented planes like one would use for grass or smaller flowers. It lets you manually place grass objects anywhere in your scene and gives you advanced bending and deferred translucency.

The shader expects bending information to be baked to vertex color alpha: Lower vertices should be set to alpha = 0.0 / higher to alpha = 1.0. RGB should be set to 1.0, 1.0, 1.0.

You can use the Grass Model Postprocessor to automatically bake the bending values on import.

In order to safe draw calls and get proper slope aware lighting you have to parent grass objects under a game object which contains the “Combine Children AFS” script as this script script will bake the underlying ground normal to the meshes. Find out more >

Shader Inputs

Albedo (RGB) Alpha (A) Main diffuse texture (RGB) Transparency (A)

Translucency (G) Occlusion (B) This slot holds the combined texture containing the translucency map in G and the occlusion map in B. You have to author this texture according to the Model and Texture Guide.
In case you are using a unified texture atlas for the terrain engine this will equal the “Combined Detail Atlas” map you assign to the “Setup Afs” script.
Please note: This texture has to be imported using “Advanced” -> “Bypass sRGB Sampling” checked.
Translucency also affects smoothness as described below.

Grass Lighting does not support direct specular reflections (due to the normals grass use to do diffuse lighting) but it supports specular ambient reflections. So in case you want to make grass a bit more grounded you may give it some smoothness.

Min Smoothness Base Smoothness of grass masked by translucency as sampled from the texture mentioned above.

Min Smoothness Max Smoothness of wet grass masked by translucency as sampled from the texture mentioned above.

Grass Shader (Critias instanced)

This shader is solely designed to be used on grass manually placed using the Critias Foliage system.
If you use this shader within the Critias Foliage system grass instances will fade out smoothly towards the given “Max Draw Distance” as specified in the Critias Foliage painter. “Foliage Type” should be set to “OTHER_GRASS”.

Shader Inputs

These are the same as in the original Grass shader.

Billboard Shader

The Billboard shader overwrites the built in one and allows you to have camera aligned billboards. Due to the gaining popularity of depth based image effects such as volumetric lighting it renders using alpha testing instead of alpha blending by default making billboards properly write to the depth buffer and play nicely with such kind of image effects.

Please note: In case you want to use the built in billboard shader you have to remove the shader from your project and restart Unity.

Shader Inputs

Will be set up by the terrain engine automatically.

Foliage shader for the terrain engine

The Foliage shader for the terrain engine overwrites the built in VertexLit shader and lets you place normal mapped, physically based shaded and procedurally animated foliage even using the terrain engine.

However the terrain engine does not allow those meshes to cast real time shadows.

Models added to the terrain engine just have a single diffuse texture. The combined "Normal (GA) Trans(R) Smoothness(B)" map is passed using the "Setup Afs" script—just as most other parameters. Find out more >

Make sure that all detail objects added to the terrain engine share the same material/texture by creating a huge texture atlas and laying out the UVs accordingly.

The shader will use the overall bending and direct and ambient lighting functions. So in case you want to also add details like small rocks to your terrain you have to add vertex colors to their meshes: Set the vertex colors RGBA to 0.0 to prevent the vertices from being bent. Vertex color alpha however may contain ambient occlusion in case you enable *"Legacy Bending"* in the AFS Setup script.

"Dry" and "healthy color" will **not** be applied to color or shade the meshes differently: As those colors are stored in vertex colors just like the bending parameters they will control bending instead:

> In case you have **enabled** *"Legacy Bending"*: Different values in the alpha channel of "dry" and "healthy color" will give you slightly differently shaded plants.
>
> In case you have **disabled** *"Legacy Bending"*: Different values in the alpha channel of "dry" and "healthy color" will give you slightly different main bending.

The shader automatically fades out instances over distance by changing the cut out value just like the grass shader does. This works pretty fine on regular plants. Solid geometry like the little rock in the demo scenes however still tends to pop in and out. You may address this issue by adding some simple upsampled noise to the rock's alpha channel (I have chosen 20% and scaled it by factor 8). Upsampled noise because the shader will most likely sample a pretty low mip level.

Translucent lighting. Due to the fact that details on the terrain do not cast any shadows translucent lighting tends to look pretty weird and way too strong. For this reason there is a global translucency multiplier just for this shader. You will find it in the "Setup Afs" script >

Please note: This shader is prepared to support single sided geometry as it uses "Cull Off". In case all plants (not grass) you add to the terrain use double sided geometry (which would be a bad idea though) you could optimize the shader by editing it and setting it to "Cull Back" instead.

Please note: This shader only supports bending completely stored in vertex colors (or legacy bending)—limited by the terrain engine which simply skips UV4 coordinates.

Please note: In case you want to use this shader along with the "grass shader for the terrain engine" you must not add any grass to the terrain as simple texture. Instead you will have model e.g. a simple quad and adjust its UVs to make them fit your combined texture atlas. Find out more >

Please note: In case you want to use the built in VertexLit shader you have to remove the shader from your project and restart Unity.

Shader Inputs

You do not setup the inputs directly. You may not even assign this shader…

So in order to configure the plants you want to add to the terrain create a material using the "regular" foliage shader and assign it to your prefab.
Then all you have to do is to add your diffuse texture atlas to **Albedo (RGB) Alpha (A)**—that's it. Leave all other properties untouched. You assign the combined texture using the "Setup Afs" script >

Grass shader for the terrain engine

The Grass shader for the terrain engine overwrites the built in Waving Grass shader and supports advanced and globally controlled, unified wind and deferred translucency.

Wind settings (except for the "Grass Tint") made in the terrain engine inspector will be ignored.

Use "Healthy" and "Dry" Color just like you are used to but please note that adjusting the alpha values of these colors will influence bending.

Please note: In case you want to use this shader along with the "foliage shader for the terrain engine" grass must not be added to the terrain as simple texture. Instead you will have model e.g. a simple quad and adjust its UVs to make them fit your combined texture atlas.
Find out more >

Shader Inputs

You do not setup the inputs directly. You may not even assign this shader…

So in order to configure the grass model you want to add to the terrain create a material using the "regular" grass shader and assign it to your prefab..
Then all you have to do is to add your diffuse texture atlas to **Albedo (RGB) Alpha (A)**—that's it. Leave all other properties untouched. You assign the combined texture using the "Setup Afs" script >

Grass Smoothness As most other values smoothness for grass on the terrain is controlled using the "Setup Afs" script >

AFS and third party packages

You may use certain AFS shaders and models with third party products like Vegetation Studio or Critias Foliage in order to benefit from AFS's advanced lighting and bending functions.

In order to do so you will always have to add the *AFS Setup Script* to your scene and set it up properly as all AFS shaders pick up wind only from that script.

Models, materials and textures just have to be set up according to AFS's specifications.

Finally do not forget to assign the AFS deferred lighting and reflection shaders in case you use deferred lighting.

Vegetation studio

AFS currently ships with one shader which is dedicated to and compatible with Vegetation Studio: The VS_AfsFoliageShader, which lets you benefit from AFS's advanced lighting and bending features even if all the placement and rendering is done by Vegetation Studio.

So in case you want Vegetation Studio to take over the rendering of all your foliage you have to assign this shader to your prefabs which then is used by Vegetation Studio and supports instanced indirect draws.

Shader Inputs

The basic shader inputs are just the same as for the regular AFS foliage shader. But you will find some new parameters listed under *VegetationStudio Specials* such as touch bending:

Use touch bend: If checked the shaders will support Vegetation Studios' touch bending.

> **Force (X) Radius (Y)** lets you determine the touch behaviour. *Force* equals the strength of the touch in relation to the displacement while *Radius* lets you define some kind of falloff. Set *Force* to your liking but keep *Radius* between 1 and 2.
>
> Overall however i have to admit that Vegetations Studio's touch bending does not allow us to derive any valuable information for more complex models such as a fern so touch bending will always look a bit odd.

Critias Foliage

In case you use Critias Foliage you can use the standard foliage shader – all you have to do is to enable *Critias Foliage Support* in the material inspector.

Known Issues and FAQ

Instancing and Lightmaps
is not supported: See Unity's docs to find out more. You may use the combine children script instead to speed up rendering.

Instancing and Light Probes
Make sure all gos you want to be drawn using instancing are affected by the same lightprobe. Do so by assigning the same "Anchor Override" (most likely the light probe itself) to all objects. – This does not work in Unity > 5.5. So foliage instances which are set to blend probes will most likely not be drawn instanced.

Instancing and forward rendering
Lights might flicker (Case 818543) Instanced surface shaders broken in forward. This should be resolved in Unity 5.4.1.

Terrain Grass
Terrain Grass Shader does not always write to depth correctly (Case 819602). This should be resolved in Unity > 5.4.1.

Using DX9 or switching back to DX11 shows shader errors
If the error points towards the Foliage shader and contains something like: "cannot be used

inside of dynamic conditional “if” blocks” – that error must be a shaderlab bug: After hitting reimport all errors are gone and in fact: the shader renders fine.

Changelog and new features

- **Basic support for Critias Foliage system** added.
- **Lighting unified and optimized for deferred rendering.** Except for grass all shaders use physically based shading built on top of Unity’s built in BRDF and support proper translucent lighting in both forward and deferred rendering.
- **GPU instancing support** for foliage and tree shaders added.
- **Wind** is now basically driven by a built in directional Wind Zone.
- **Foliage shader optimized for single sided geometry** which lets you save a lot of vertices and still gives your correct lighting in both forward and deferred rendering.
- **Foliage shader for the terrain engine** high quality tangents added (instead of the formally estimated ones) so bump mapping works perfectly :)
- **Foliage shader now supports proper bending driven by vertex colors only**. Baked ambient occlusion is not supported in this mode – instead you get full control over bending while keeping the amount of data per vertex at a minimum. **Please note:** Legacy bending will be dropped with the next release.
- **Foliage shader** lets you adjust the view dependency for translucent lighting.
- **Tree shaders** use the new unified lighting which means physically based shading and deferred translucency. Transitions between mesh and billboard are improved as shadows will fade in. Added support for wetness (beta).
- **Billboards** now use alpha testing and write to the depth buffer properly by default in order to make them work with depth based image effects like volumetric lighting. Alpha blending might be enabled manually.
- **Grass shaders** now support wetness (beta).
- **Touch bending revisited.** No need to use different materials and shaders anymore. Simply enable touch bending in the material editor and add the “touchBendingCollision” script to your model which now handles a (theoretically) endless number of registered touches.
- **Foliage Tool** now supports Blueprints which let you copy settings from prefab to prefab in order to e.g. create LODs. Improved adjustment of vertex colors and Debug View added.
- **Wind on grass** has been reworked. Please check your scenes and adjust the “Wind settings for Grass Shaders” properly.
- **Combined Meshes** Proper information about the original pivot now are baked as well so you will get less stretching and advanced variation in bending.
- **Setup Script** some performance improvements added.
- **Dropped support for static batching** Use GPU instancing instead or combine instances using the Combine Children script.
- **Formally statically combined and baked meshes** may be corrupted.

- **Tree shaders** Dropped support for billboarded leaf meshes for performance reasons. I guess pretty much nobody uses that feature anyway. But in case you do you may reactivate it by editing the shader.