V 2.04

Advanced Foliage Shader

2

The advanced foliage shader gives you the possibility to use all modern features of vegetation shading on imported, handmade models – namely procedural bending, bumpmapping, specular lighting, translucency and baked ambient occlusion.
These features are well known from the built in tree creator trees but now you can use them with any of your own models after having done some tweaks.

Additionally v > 2.0 comes with additional features such as touch bending which will make your plants bend when touched by the player, npc or any other game object or ground based lighting.

So let’s get started.

Basics of the afs

Before I will describe how to tweak your models I would like to make you familiar with the basic techniques of foliage shading in order to give you a deeper understanding of what is going on, so you can perfectly adjust all necessary parameters.
Convincing foliage rendering consists of two parts: shading or lighting which is aiming its surface and bending which is just... bending.

The advanced foliage shader is a simplified version of the built in tree creator shaders which in turn are based on Crytek's foliage shaders (see: http://http.developer.nvidia.com/GPUGems3/gpugems3_ch16.html).
It supports the original lighting function (which is wrapped around diffuse plus translucency) but comes with a slightly modified bending function as the original one makes it quite difficult to adjust your models to its needs.
As shading/lighting should be rather clear to anybody familiar with the built in tree creator I will just focus on bending.

Bending

Crytek's implementation of bending consists of three blended animations:
1. Main bending, which animates the entire model along the wind direction.
2. Detail bending, which animates the branches and leaves – mostly in along the y-axis.
3. Edge fluttering.

In order to make the whole blending more believable, there is even a fourth factor:
4. per-leaf / per-branch phase variation.

All these four factors are controlled by vertex colors of which we have also four: rgba.
Unfortunately we also would like have ambient occlusion baked in (vertex alpha) so we just have three colors left for four params...

Differences between afs and the built in tree creator shaders

In contrast to the built in shaders, which use vertex colors and a second uv set to solve this conflict, the afs just combines main and detail bending into one parameter so all bending is just controlled by three vertex colors: rgb. Of course this ends up in some limitations as far as the accuracy of bending is concerned but this was the lesser evil as main bending is mostly important for larger geometry with solid structures like trees.

Pros and Cons

The effect of this limitation gets visible at the banana tree model: Although the trunk is rather thick and stiff it not only bends in the wind's direction but also pulses a little bit and moves up and down... not exactly what we want, but quite ok I think – at least as the banana is more or less a tree rather than a bush or shrub and should may be use a different shader...

On the other hand afs gives you some advantages which should not be unmentioned like batching – next to the fact that modifying your models to the needs of afs is much more easy than making them fit the needs of the built in tree creator shaders of course.

But back on track. What does it need to use your models with the advanced foliage shader?

Basic Requirements

Geometry

In order to support (receiving and casting) shadows we need double sided geometry on anything that is just a single plane like some kind of leaf, for single sided geometry would not be written to the shadow map correctly and could not be lit in a proper way.

Vertex Colors

As all bending is controlled by vertex colors, make sure your models have vertex colors according to the requirements described below.

Textures

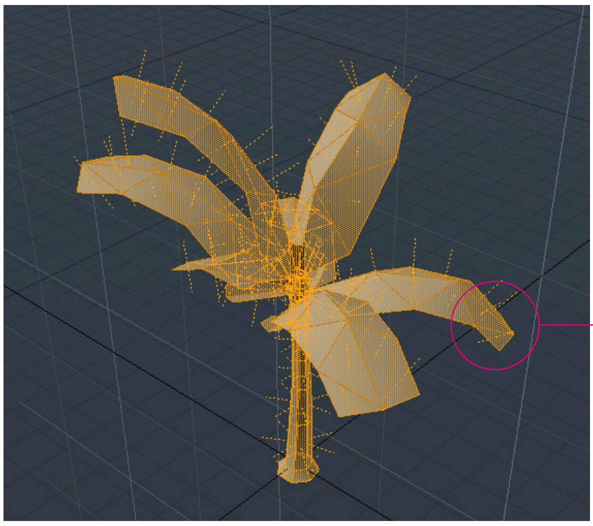
The afs supports diffuse, alpha, normal, specular and translucency maps but in order to keep the texture load and bandwidth footprint as small as possible it works with just two combined textures: diffuse/alpha and normal/specular/translucency.

Wind

Unity’s built in wind zones are not accessible via script that easily. So we will have to create our own wind and provide it to the shader, where wind consists of direction (xyz) and strength. The package ships with a simple script that animates wind and pushes it to the shader.

Adjusting your models

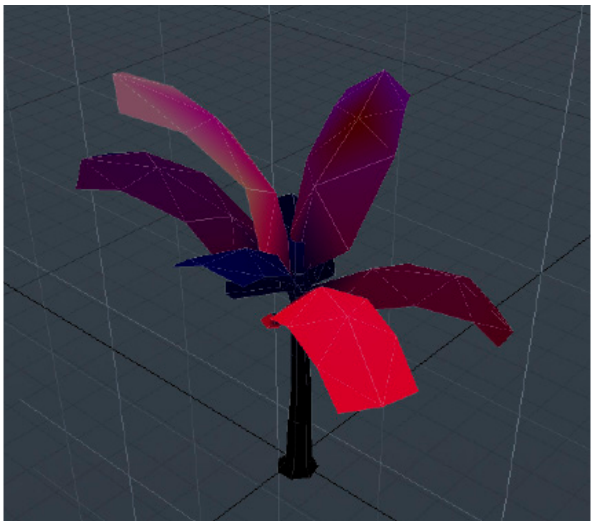
Geometry



Just model your model the way you like and unwrap it. Then select all single, simple planes for leaves or whatever, duplicate them, flip their normals and merge them with the original mesh. In the end we need one single mesh with just one material assigned to it.

Leaves planes must be double sided.

Vertex Colors



This is the most challenging part of all.
As mentioned before bending is controlled by four parameters which are:

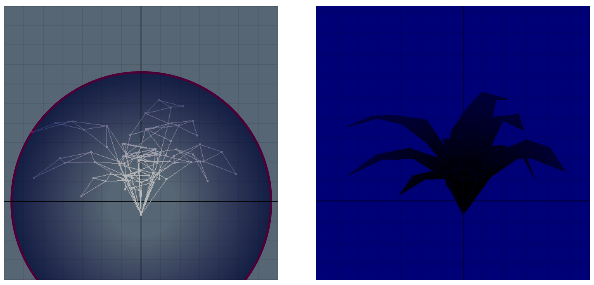
1. Main bending	-->	Vertex color blue
2. Detail bending	-->	Vertex color blue
3. Edge fluttering	-->	Vertex color green
4. Phase variation	-->	Vertex color red

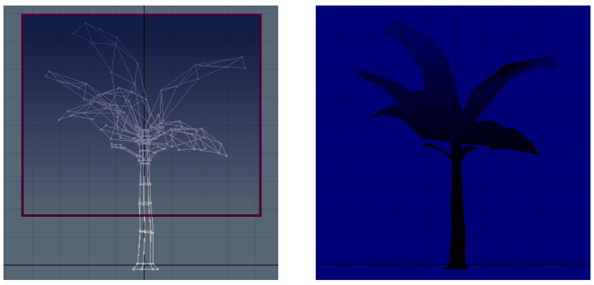
Adjusting main and detail bending

As edge fluttering and phase variation are just two nice add ons I would highly recommend to start with adjusting main and detail bending first.
Doing so 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 values of vertex blue 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.
***New in v. 2.03:** Just set up edge fluttering and phase variation using your 3d app – and use the AFS Foliage Tool to set up primary and secondary bending right within unity.*

> ***Tip:*** *When starting adding vertex colors for main and detail bending to the 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.*

The way how to apply vertex color blue 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 the most complex one, for it consists of a rather stiff trunk and pretty flexible leaves.



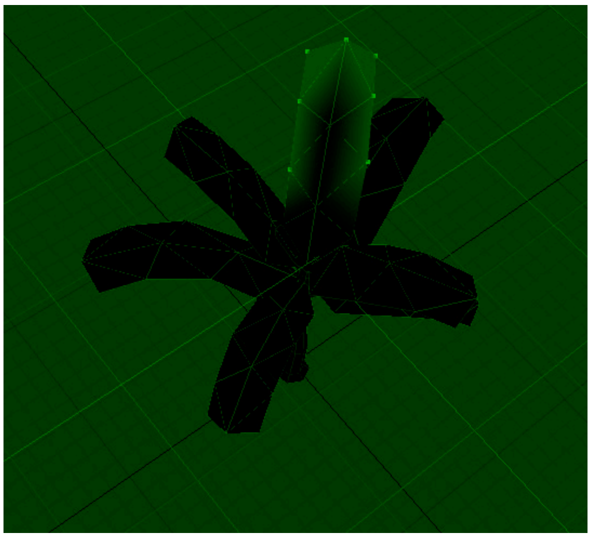


Setting up the fern model
As we do not want to have any bending at its pivot but a lot of it at its outer leaves we simply assign a radial gradient from blue=0 at its pivot to blue = 0.5 at its outer regions.

Setting up the banana tree
Simply assign a linear gradient from blue=0 at one third from its pivot to blue = 0.7 at its upper parts.

That is all for the moment. Just bring those two models into Unity and compare their bending to each other. You might want to give the fern a little less bending, the banana tree a little more or vice versa.

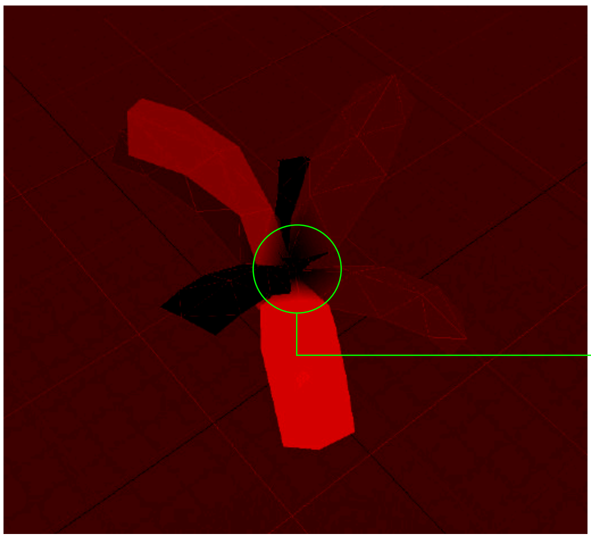
Adjusting edge fluttering



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.*

Adjusting phase variation



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

~~***Note:*** *Vertex color red will also be multiplied to the color value of the diffuse texture in order to provide more variety in base coloring of the leaves.*~~ *Dropped in v. 2.0.*

Tip: *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.*

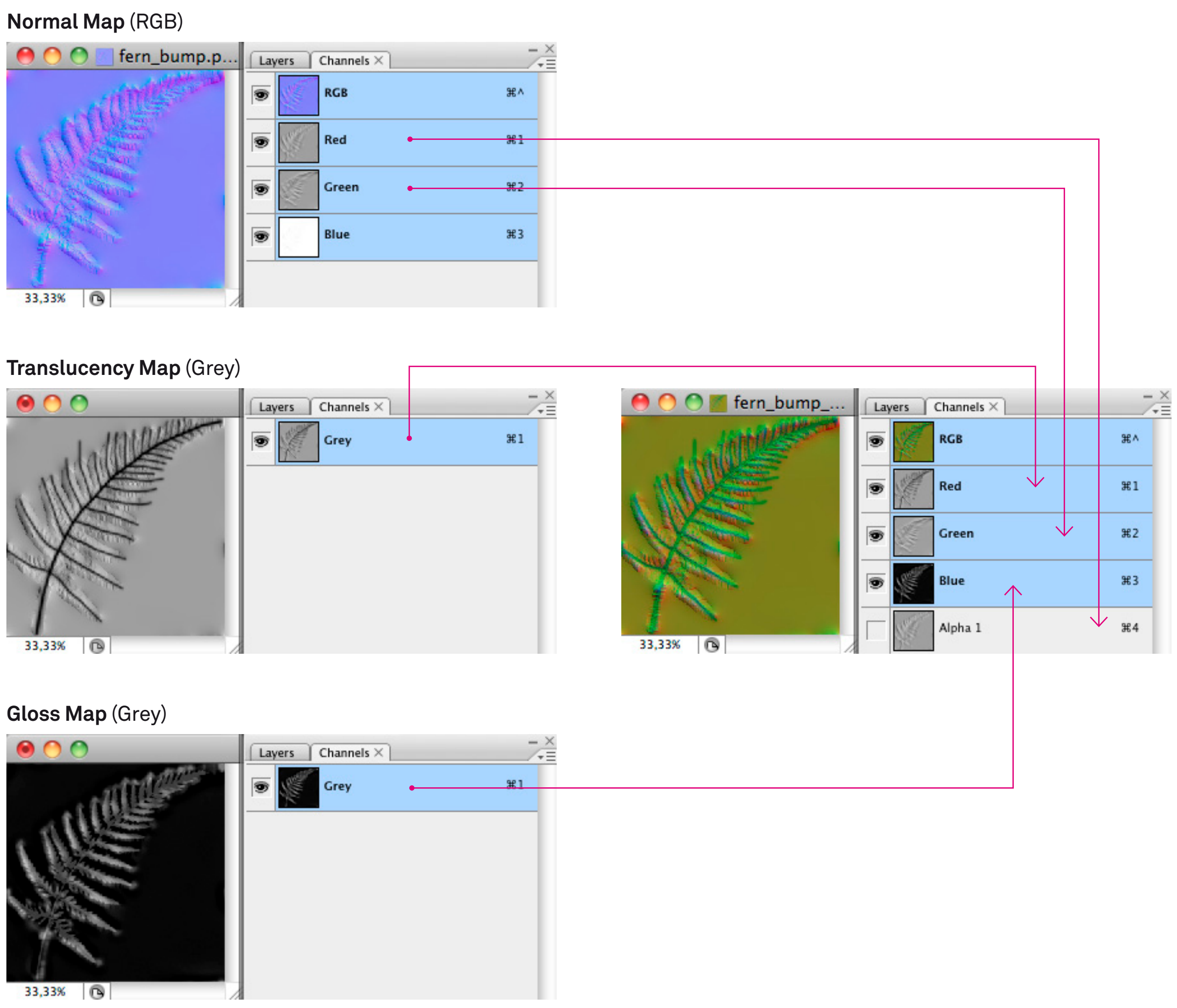
New in v. 2.0: *In case you use touch bending vertex color red is added on top of vertex color blue in order to give you more variety in touch bending. For this reason i would recommend to lower the range of uses red tones to e.g. 0.5 – 1.0. Please note: Branches which are pretty much exposed and should be strongly effected by touch bending should have high red values.*

Adding ambient occlusion

Ambient occlusion can be baked to vertex color alpha. 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
And here is a short video showing it in action:
http://www.youtube.com/watch?v=eeKQAXg-Qo8

Setting up the normal/specular/translucency texture

As the shader needs a single combined texture for normal/specular/translucency you will have to generate this texture manually like shown in the figure below.



Wind

Although you might already have a wind zone in your scene, you will not see any bending on any model using the advanced foliage shader unless you also add the setupFoliageShader script, for wind zones only affect trees modelled using the tree creator.
The setupFoliageShader script just provides a very simple setup and animation for your wind and lets you set the wind direction (x, y, z), strength (w) and its frequency.

Next to providing the shader with the needed wind parameters this script also helps to optimize performance by setting up culling layers. So you might cull all manually placed ferns at exact the same distance as all detail meshes placed within the terrain engine are culled and cull all bananas at another distance.

Last but not least it pushes all needed parameters to shader which are needed in case you use the shader within the terrain engine.

See the added .rtf files for further details.

Using the shader within the terrain engine

In case you want to draw foliage even faster you can use a special tweaked version of the foliage shader within the terrain engine (advancedFoliageShader for Terrainengine.shader) which will replace the built in vertexLit shader.

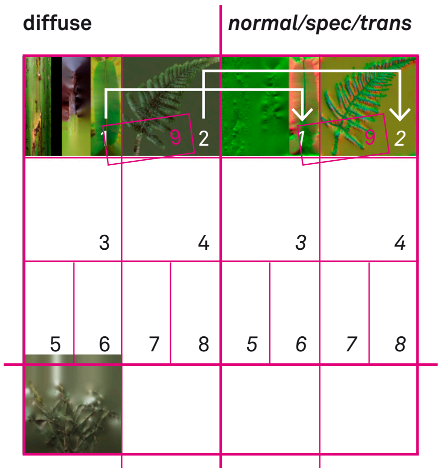
Please note: *Placing meshes within the terrain engine using the shader implies all disadvantages placing meshes within the terrain engine usually implies: they do not cast shadows and they do not have any collider – which might be bad at least for our banana tree.*

Due to the fact that the terrain engine does not pass any tangents to the shader, tangents have to be calculated on the fly which unfortunately leads to slightly different lighting.

In addition to this you will have to create a texture atlas including all meshes you would like to use within the terrain engine and map your models to this atlas. So you might need other models than those placed manually – although you can also use the models made for the terrain engine as single game objects of course.

Please note: *Using a manually generated atlas won't let you use any grass rendered as billboard or simple texture, because adding this kind of grass would add a second texture to the internal texture atlas of the terrain engine and corrupt the texture lookup within the manually setup atlas.*

Setup of the texture atlas



If you use the **ats v2 waving grass** shader for the terrain engine all grass textures have to be placed in the bottom row – otherwise bending will not work correctly.
If you use the **ats v3 waving grass shader** this limitation does not exist.

Setting up the texture atlas
Make sure that the texture atlas has a square size.
Place all diffuse textures in the left half, all normal/spec/trans textures in the right half of the atlas.

Of course you do not have to use square textures within the atlas as the image on the left might suggest. It is just ok to make sure that the x-offset of the normal/spec/trans map in the right half matches the x-offset of the diffuse texture, as the shader just looks up the diffuse texture coordinates and adds 0.5 to find the normal/spec/trans map. Y coordinates do have to fit in any case.

Using the ats waving grass shader V2
In case you are using the ats waving grass shader V2 /which replaces the built in grass shader) all grass textures (textures for meshes using the grass shader) must be placed in the bottom row of the atlas for the bending of the ats grass shader is based on the uv's v coordinate.

Using the ats waving grass shader V3
The ats waving grass shader V3 doesn't come with any restrictions as far as the uvs are concerned.

Tip: *Unity's default import settings do not allow textures to be more than 1024 px in square. For this reason make sure that the import settings match your atlas' size (max is 4096px).*

Adding meshes to the terrain

Simply add your mesh using the “add detail mesh” function and choose “Vertex Lit” as render mode in the upcoming dialog. Use “Healthy” and “Dry Color” to add more variety in bending to your instances: Blue will change the overall bending, Green will add more edge fluttering , Red will change the animation phase and Alpha will darken or brighten the instance. Just tweak it to your needs.

ATS Waving Grass Shader v3 for Terrain Engine

This shader lets you place even complex single sided geometry as grass within the terrain engine and will stop it from floating around unlike the built in shader as it uses vertex color alpha values to control the amount of bending.
All you have to do is to add some simple vertex colors to your meshes: Vertex color alpha = 0 means no bending and should be applied to the lower vertices whereas vertex color alpha = 1 means full bending which you will probably apply to the upper vertices.
Using vertex colors instead of uv ccordinates sets you free to position the grass textures whereever you want within the texture atlas and lets you even create complex shapes like 3 dimensional flowers or small saplings with overlapping uvs.

Touch Bending and other new Features in v2

Please have a lok at the attached .rtf files to find out more.