Lux Instant Vegetation

Thanks for downloading this package.

In order to get you started right away please go to Getting Started.

If you are looking for specific shader inputs you will be covered in the chapter Shader Inputs.

Overview

Lux Instant Vegetation consists of a set of shaders which add high quality procedural wind animations to pretty much any tree or bush mesh - almost instantly and without the need to bake bending information.

In case you want to improve the final result the shaders also support baked bending information from a bunch of different sources.

Getting started

In order to start I highly recommend opening the included demo scene and studying the provided examples as these will give you an idea of which param needs which value.

Geometry requirements

The shaders expect upright oriented trees (in object space). Make sure the model's scale is set to 1.0 and has no rotation in your DCC app.

Step by step

1. If you start from scratch make sure your scene contains a **Unity Wind Zone** and the **LuxInstantWind.cs** script is attached to it. This script samples the wind settings from the wind zone and passes these to the shaders.

   Start with moderate values in the wind zone like Main = 1.0 and Pulse Magnitude = 0.5.

2. Drag your model into the scene, create a single material using one of the LuxInstantWind shaders and assign it to ***all*** sub meshes. I personally use the Debug shader as it lets you debug e.g. the wind mask.

   Of course later you will most likely use at least 2 different materials for bark and leaves but in order to start it is handy to only have one single material as you will have to tweak a lot of parameters. In case you had 2 materials you would have to keep these in sync all the time…

3. Start tweaking the *Main Wind*, *Main Power* and *Speed* parameters which will add main bending to your model.
4. Then check *Enable Branch Wind* and start setting it up. While adjusting the procedural *Wind Mask*, its *Falloff* and the *Directional Mask* you may just go nuts regarding the strength and speed params. In fact this may even help getting a better picture of the final mask.
5. Next check *Enable procedural Phase* and set the *Procedural Tiling*.

   Small values around 0.05 should be a good starting point. Larger values will add a higher frequent noise and most likely add too much distortion to your model.

6. Now you can fine tune the branch wind settings and re-adjust the wind mask.
7. Finally *Enable Edge Flutter* but note: *This feature needs a proper mask baked into the vertex colors to avoid any disconnections between leaf planes and branches.*

   It should be pretty easy to add the needed vertex colors: Just open the tree or bush in your DCC app, select all the leaves, isolate them and flood them with vertex color green = 0.0. Then switch to the UV panel select all outer vertices but the ones which connect the leaf planes with the branches and set them to e.g. Vertex color green = 1. Vertex color green is used by most shaders to mask edge fluttering.

8. Finally create a 2nd material and copy the wind settings from the one you set up.

Please note: Wind settings of all materials of a tree have to match!

Bending

Bending is composed out of 3 components: Main bending, branch bending and edge flutter.

Main bending

Main bending bends the entire tree along the wind direction based on the y coordinate of the given vertex. Incoming wind from the wind zone is animated within the shader based on the *Speed* parameter of the shader and the pivot's position in world space. The final strength is controlled by *Main Bending* and *Main Power* which work closely together.

Branch Bending

Branch Bending usually is what a tree makes look like a tree. In case you go fully procedural however this most likely will never look fully convincing - no matter how many noise samples we add on top of each other.

Branch bending can be controlled by baked vertex colors although only a few models have these applied.

So the shader offers just a simple **procedural approach** which consists of:

- a cylindrical or hemispherical ***Procedural Wind Mask*** which prevents mainly the trunk from being affected by the Branch Wind
- and the ***Procedural Phase*** parameter which modulates the *phase* of the branch bending in 3D space based on the ***Procedural Tiling*** and a simple sine based function. This "3D noise" of course has no knowledge of the tree structure and may result in some nasty distortion.

In case you want to spend some time on your models you can tell the shader about the tree structure by **painting the mask and phase** into vertex colors.

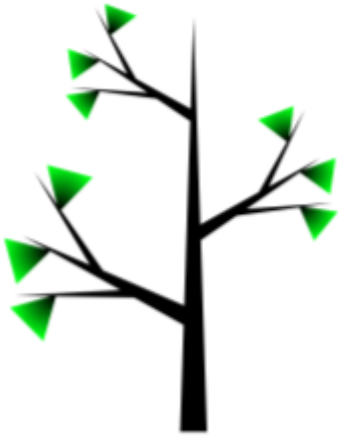
As painting the mask is quite cumbersome and not as important as the phase I will skip it here. Painting the phase however is a doable task. The preferred color channel would be red. Simply color the main branches including all their sub branches and leaf planes with unique shades of red and you will get a more believable bending and no distortion.

The branch bending animation then consists of *Displacement* - a movement up and down - and *Rolling* which rotates the branches around the y axis.

Phase in VertexColor.Red
Note: Define "systems" starting with the first branch which is directly connected to the trunk. Then apply the same red value to all connected sub branches and leaf planes.

Edge Flutter

Edge Flutter is a high frequency animation specially for leaves **which needs you to edit the tree** and manually paint some vertex colors as a mask in order to avoid disconnections between branches and leaf planes. The preferred color channel is green.



Flutter in VertexColor.Green
Note: Make sure that the vertices which connect the leaf planes to the real branches have VertexColor.Green = 0.

LuxInstantWind script

This script is needed to grab wind settings from the Unity wind zone and push these to the shaders. It has to be added to the game object holding the wind zone component.

Inputs

Main Wind, **Turbulence** and **Pulse Magnitude** are simple multipliers for the values coming from the wind zone and let you globally tweak the bending.

Map Main To Turbulence If checked the script will skip the Turbulence multiplier and derive *Turbulence* from *Main* according to the *Main To Turbulence* animation curve. This frees you from adjusting turbulence manually and automatically clamps max turbulence - which otherwise just may go nuts.

Main To Turbulence Lets you specify how to map main to turbulence.

Wind Fade Distance The distance at which wind calculations will be skipped in the vertex shaders.

Shader Inputs

Base Inputs

I won't go into details here as it is just regular stuff. And as all this is exposed in the Shader Graphs you may easily tweak it to fit your needs - like using better packed textures.

Only thing worth mentioning: The "Mask" texture in the leaf shader expects "Thickness" in the blue color channel - instead of the detail texturing mask like in the original HDRP mask map layout.

URP Lighting

As the name suggests this category is only available in the URP Leaf Shader.

Flip Backface Normals If checked the shader will flip the normals of backfacing triangles - which is mandatory for proper physically based shading and will prevent false specular highlights. However if the model uses smoothed normals this might fully corrupt lighting.

NOTE: You could skip Flip Backface Normals and disable Transmission which will end in a faster shader but may give you a lot of false specular highlights.

Enable Transmission If checked the shader will add transmissive lighting. This will be output to the Emission node and only be calculated for the main directional light. Additional point or spot lights are not supported.

Strength(X) Power(Y) Distortion(Z) *Strength* should be easy to understand, *Power* here drives the view dependency of the transmissive lighting (smaller numbers makes it less view dependent) *Distortion* lets you distort the vertex normal used to calculate the final strength which adds some variation here.

Ambient Scattering If > 0.0 the shader will add transmissive lighting from the ambient lighting as well.

Wind

Main Bending Strength of the Main Bending along the wind direction.

Main Power Controls the stiffness of the trunk. Smaller values will make it stiffer.

Scale along XZ Values > 0.0 will add some more main bending to outer vertices.

Speed Speed of the main bending animation.

Enable Branch Bending If checked the shader will calculate secondary branch bending.

Enable baked Wind Mask If checked the shader will pick the wind mask from the vertex color channel defined below.

Mask VC Channel The vertex color channel which stores the wind mask. *0 means red, 1 means green, 2 means blue and 3 means alpha. Please use integers here!*

Procedural Wind Mask Radius of the procedural mask which masks out branch bending (especially on the trunk).

Use Hemisphere If checked the wind mask is calculated using a hemispherical falloff. By default it uses a cylindrical falloff oriented along the up axis.

Scale Lets you scale or skew the cylindrical or hemispherical mask. *Needed for small objects like ferns. Here a value of 6, 4, 6 might fit.*

Falloff Falloff of the mask.

Directional Mask Lets you mask out wind on the side pointing away from the wind (“wind shadow”).

Branch Wind Speed Speed of the branch bending animation.

Rolling Strength of the rotation around the y axis. *If Rolling = 0.0 the shader will skip some computations.*

Displacement Strength of the movement up and down.

Enable baked Phase If checked the shader will grab the phase from the vertex color channel defined below.

Phase VC Channel The vertex color channel which stores the phase. *0 means red, 1 means green, 2 means blue and 3 means alpha. Please use integers here!*

Enable procedural Phase If checked the shader will calculate the phase based on a simple sine function.

Procedural Tiling Tiling or frequency of the procedural phase in object space.

Procedural Strength Final multiplier on the branch wind.

Enable Edge Flutter If checked the shader will apply edge flutter. *Please note that edge flutter needs proper vertex colors being provided. See chapter “Wind”.*

Flutter Mask VC Channel Lets you choose the vertex color channel in which your flutter mask is stored. *0 means red, 1 means green, 2 means blue and 3 means alpha. Please use integers here!*

TVE Flutter Mask Override If checked the shader will expect a TVE baked flutter mask in UV0.z. *This will override the Flutter Mask VC Channel settings above.*

Flutter Speed Speed of the edge flutter animation.

Flutter Tiling Tiling or frequency of the edge flutter.

Bark shader

The bark shader also has the Edge Flutter parameter - which does not make much sense but is needed to make the shader compile. So simply ignore Edge Flutter here and disable it in the bark material.

Debug shader

The debug shader lets you debug the calculated wind mask or procedurally generated phases. As Shader Graph does not support enums, features here are implemented using booleans. The last checked boolean will win.

Common vertex colors usage

Tree Creator

Red Variation or phase
Green Edge flutter mask
Alpha Ambient Occlusion

TreeIt

Red Edge flutter mask
Green Variation or phase
Blue Wind Mask

CTI

Red Variation or phase
Green Edge flutter mask
Alpha Ambient Occlusion
Wind Mask unfortunately is stored in UV1.

The Vegetation Engine

Red Variation or phase
Green Wind Mask
Blue AO
Edge flutter mask is encoded into UV0.z

Nature Manufacturer

Alpha Edge flutter mask