

Conifers [BOTD]

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 Properties](#).

Do not miss the chapter [Optimizations](#).

Table of Content

[Conifers \[BOTD\]](#)

[Table of Content](#)

[Getting Started](#)

[Built in render pipeline](#)

[Deferred Rendering](#)

[Vegetation Studio Pro](#)

[URP](#)

[HDRP](#)

[Technical Notes](#)

[LODs](#)

[Wind](#)

[Materials](#)

[Textures and Texture Arrays](#)

[Optimizations](#)

[Bandwidth and Memory](#)

[Fillrate](#)

[Further optimizations](#)

[Shaders and Shader Properties](#)

[CTI/Bark and CTI/Bark Array Shader](#)

[Shader Inputs](#)

[CTI/LOD Leaf Shader](#)

[Shadows](#)

[Lighting](#)

[Wind](#)

[CTI/LOD Billboard shader](#)

[Shader inputs](#)

Getting Started

Right after having imported the package the automatically launched “Installer” window lets you choose the proper render pipeline support package which contains the needed shaders, materials, prefabs and supporting assets.

Once you have installed all needed sub packages you are ready to explore the trees. I highly recommend starting with one of the provided demo scenes located in “Render Pipeline Support” -> “Your Pipeline” -> “Demos”.

Built in render pipeline

In order to get the best visual results please make sure that your project is set to use the **linear color space** in:

Edit → Project Settings → Player.

Deferred Rendering

In case your camera uses **deferred rendering** you have to assign the **CTI deferred lighting shaders** in:

Edit → Project Settings → Graphics:

- Under the *Built-in shader settings* change *Deferred* to *custom*, then assign the *CTI_Internal-DeferredShading* shader.
- Also change *Deferred Reflections* to *Custom* and assign the *CTI_Internal-DeferredReflections* shader.

Otherwise the trees will look pretty colorful...

Other deferred lighting and reflection shaders which should work out of the box are those of Lux Plus, AFS and ATG.

Vegetation Studio Pro

As the (customized) included CTI shaders identify as “xyz 301” Vegetation Studio Pro will not recognize them as CTI shaders.

Fix this by importing the **VSPPro_CTIShaderController.package**.

URP

In case you use URP you can right jump into the demos.

HDRP

In case you use HDRP you can right jump into the demos but might spot some pinkish color on the trees as the provided **CTITreesDiffusionProfile** has to be added to the pipeline settings. Do so by editing the material, scroll down to the Diffusion profiles and hit “Fix”.

Technical Notes

LODs

All trees use Unity's built in LODGroup component. If you enter play mode and move around in the scene you will get a first impression of how the **LOD blending** looks like: It pretty much fits the Speedtree 7 blending: It fades out specified leaf planes during the transition of the mesh LODs and performs a dithered crossfading between the last mesh LOD and the billboard.

Please note: Proper LOD blending may only work in play mode.

While leaves are animated during the transition branches and any simplification made to the geometry simply pop. In case you use **temporal anti aliasing** this will already smooth out this popping a bit. In order to get an even smoother transition you may try to change the LOD fading from **Speed Tree** to **Cross Fade**. This however will draw each tree **twice** during the transition and most likely be pretty costly.

LOD0 is the original modeled mesh – while LOD1 and LOD2 are pretty rough simplified versions: Here branches may get disconnected from the trunk and you will notice a lot of texture stretching. Not noticeable at a common LOD distance so I did not care about this. On the other hand it does not let you replace LOD0 with LOD1.

Wind

Entering play mode you will also see the wind in action. It is driven by a standard Unity wind zone which also holds the *CTI Custom Wind* script: As our trees (at least when it comes to URP or HDRP) do not contain a "Tree" script component we have to grab the wind settings from the zone and push these down to the shaders by script.

Please note: Only one directional wind zone is supported. Radial wind zones are not.

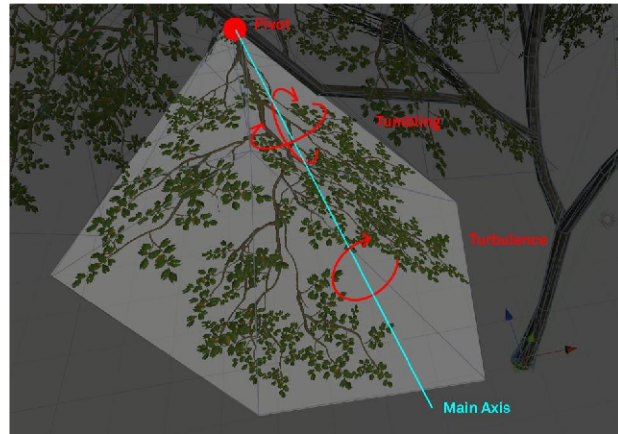
Rotating the wind zone or changing e.g. its *Main* parameter will give you instant feedback on the trees. Their bending animation is driven by direction, strength and turbulence coming from the wind zone. The shaders translate this information into several bending animations which gets calculated and mixed in the vertex shader:

- **Main bending** Bending of the entire tree along the wind direction.
- **Branch bending** Bending of the branches and leaves.
- **Edge flutter** high frequent bending of the outer edges of the leaf planes.

Please note: When changing the "Base Wind" -> "Multipliers" on the leaf material you will also have to tweak the settings of the bark and billboard material.

On top of this pretty common tree bending invented by Crytek and used e.g. by the built in tree creator shaders the CTI shaders support:

- **Leaf Tumbling** rotates each leaf plane around its original pivot according to the wind's direction and strength.
- **Leaf Turbulence** will rotate each leaf plane around its original pivot according to the main axis of the leaf plane.
- **Advanced edge bending** will add a periodic wave animation along the main axis of the leaf plane.



Please note: Leaf tumbling and turbulence need additional information being baked into the mesh giving you a “fat vertex”. They also need more ALUs and thus will be less performant than just the base wind animation. Consider disabling advanced wind features in case you have performance problems.

Materials

When using the built in render pipeline **GPU Instancing** should be enabled in the material inspector in order to draw trees efficiently. URP and HDRP both rely on the SRP Batcher if you place the trees manually. In case you use the terrain engine even URP and HDRP may utilize GPU Instancing .

Textures and Texture Arrays

The bark material usually uses the “Bark Array” shader by default which allows you to use 2 different texture sets, stored in 2 texture arrays. So the lower parts of the trunks use a unique texture set while the upper parts of the trunk and the branches use a generic tiling texture set.

Texture arrays typically are provided in 0.5/1K, 1/2K and 2/4K resolutions as you may not be able to create texture arrays your own. In order to do so I recommend getting the free [Texture Array Inspector](#) from the asset store.

In case you do not care about the lower parts of the trunks being uniquely textured you may change the shader to the “Bark”. This will help to save some bandwidth.

Please note: Texture arrays do not respect the “Global Mipmap Limit” as defined in the quality settings but will always use all mips.

Optimizations

Trees are expensive to render as leaves usually produce a lot of overdraw putting a lot of pressure on the raster units of the GPU (fill rate) and the memory bandwidth. At least the latter can easily be addressed.

Bandwidth and Memory

CTI already use highly optimized combined textures but actually we can do even better:

- Use the lowest texture resolution feasible – especially on all leaf textures.

- **BIRP:** Use a **dedicated Shadow Map Alpha** texture in the shadow caster pass: Doing so the GPU does not have to sample a 21.3 MB 4K RGBA texture but only a 10.7 MB 4K RGB texture (BC4 compressed). In case you use **deferred rendering** you can even think about lowering the resolution of the *Shadow Map Alpha* texture from e.g. 4K to just 1K. Actually I use a texture resolution of only 256 by 256 pixels. **In case you use forward rendering please make sure that the import settings for the Shadow Map match those of the albedo texture (mip maps, wrap mode, filter mode, aniso level) as otherwise you may get gaps in the shadows and depth buffer.**
- **HDRP:** Use a **dedicated Alpha Map** and a simple *RGB Albedo* texture on the leaf material - as during the Depth Prepass and the Shadow Caster Pass only alpha is needed whereas in the GBuffer pass the shader will only sample Albedo (RGB).
- Use **different resolutions** on the different textures: The combined AO/Translucency/Smoothness texture may not need 4K while the Albedo texture does.
- In case you do not need a unique texture set on the roots (because it is hidden by other foliage such as fern anyway or whatever reason else) simply **skip the bark array shader** and use the regular bark shader instead: This will cut the memory needed for bark by half.

Fillrate

- **BIRP:** Try to skip back faces in the shadow caster pass. Do so by setting *Shadow Caster Culling* to *Back*. **In case you use forward rendering Shadow Caster Culling should be set to Off** tho as the shadow caster pass is also used to render the trees during the depth pass.

Further optimizations

- Deactivate Animations like tumbling, turbulence or advanced edge flutter if you think your game can do without these.
- **BIRP:** Setting "Ambient Scattering" to 0.0 will make the pixel shader skip a 2nd lookup of the ambient lighting.
- Tweak the LOD settings and make sure that the billboards kick in as early as possible.

Shaders and Shader Properties

The bending of the trees has been created using the *Custom Tree Importer* (CTI for short) and thus the trees use the CTI shaders. CTI supports the BIRP, URP and HDRP – however shader inputs may differ in the different render pipelines.

Please take a look into the .pdf files in the CTI runtime folder of the different render pipelines in case you come across a parameter not mentioned here.

CTI/Bark and CTI/Bark Array Shader

Shader Inputs

Color Variation Trees will be slightly different tinted according to their position in world space. RGB defines the tint color, alpha the strength of the tint. *Always make sure that all shaders (leaves, bark and billboard) share the same color variation values.*

Albedo (RGB) Smoothness (A) Diffuse texture which contains **smoothness** (unlike the leaf shader which expects transparency) in the alpha channel.

In case you use the Array shader the shader expects a texture array with 2 different textures.

Normal Map (GA) Specular (R) AO (B) contains the combined normal, specular and ambient occlusion map. The channel layout of this texture differs from that of the leaf shader.

In case you use the Array shader the shader expects a texture array with 2 different textures.

R = Specular as simply grayscale as dielectric materials do not have a colored specular. Gets compressed pretty lossy... . You can specify different shades of gray in your texture but please note that the associated billboard shader simply uses a simple spec color value.

Please note: Spec color here is **linear**, so Unity's default 51,51,51 is way too bright. It should be RGB = 6,6,6 – but that seems rather dark. RGB around 10,10,10 however looks fine.

G = Green channel of the regular normal map

B = Ambient Occlusion

A = Red channel of the regular normal map

Please note: As this texture is **NOT** a regular diffuse texture nor a normal map you have to switch its import settings to "Advanced" and check "ByPass sRGB Sampling" – or uncheck "sRGB (Color Texture)". Trilinear filtering is recommended.

Secondary Maps (need UV2) (BIRP only) In case you have created UV2 and made the importer to add these, this drop down lets you specify how the shader should handle the secondary maps.

The trees in this package have no UV2 set.

- **Disabled:** Secondary maps will simply be ignored.
- **Enabled:** Secondary maps will always be rendered. Use this e.g. on the material assigned to LOD00.
- **Fade Base Textures:** The base textures will be faded out towards the LOD switch. Use this e.g. on LOD01. Then use *Skip Base Textures* on LOD02.
- **Skip Base Textures:** Base textures will be totally skipped and only the secondary maps will be sampled and applied. Use this e.g. on LOD02.

If you setup and assign the materials for the different LODs like described above LOD00 will use base and secondary textures, LOD01 will do too but fade out the base textures, while LOD03 will only use the secondary maps.

Of course you can already fade out the base textures towards LOD01.

Swap UVS lets you swap UV0 and UV2.

Average Color (RGB) Smoothness (A) In case the shader skips the base textures on higher LODs, it still should add some "average" color and smoothness from the base texture in order to make the final result match the result from the LOD before to get

smooth transitions. You may set this color manually or simply use the button “Get average color” at the bottom of the inspector.

Detail Albedo x2 (RGB) Smoothness (A) Secondary albedo and smoothness texture.

Normal Map (GA) Specular (R) AO (B) Secondary combined normal, specular and occlusion map.

Normal Strength Lets you adjust the strength of the secondary normal.

Wind Multipliers (XYZ) Multipliers for the baked bending strengths

- **X** main bending
- **Y** secondary or branch bending
- **Z** edge flutter

Please note that these multipliers have to be synced across bark and leaf material as otherwise leaves will lose their connection.

Use Wind from Script (BIRP only) In case you place your LOD trees using the terrain engine you have to check this in order to make the trees receive at least proper directional wind. Must be the same for leaves and bark.

Fade out wind (BIRP only) In case your billboards do not use wind you may create an extra material just for the *last LOD* using mesh trees, assign it to the corresponding LOD and check this feature.

CTI/LOD Leaf Shader

Culling Use “Off” in case you use single sided geometry (as the provided trees do). “Back” would be the correct choice for double sided geometry. “Front” is available just because it would be the third possibility...

Color Variation Trees will be slightly different tinted according to their position in world space. RGB defines the tint color, alpha the strength of the tint. Always make sure that all shaders (leaves, bark and billboard) share the same color variation values.

Albedo (RGB) Alpha (A) Diffuse texture which contains transparency in the alpha channel.

HDRP uses a separate Alpha map which stores alpha in the red color channel.

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.

Normalmap (GA) Specular (B) contains the combined normal and specular map. Its channels should be set up like this:

R = unused (should be set to black)

G = Green channel of the regular normal map

B = Specular as simply grayscale as dielectric materials do not have a colored specular. You can specify different shades of gray in your texture but please note that the associated billboard shader simply uses a simple spec color value.

Please note: Spec color here is **linear**, so Unity’s default 51,51,51 is way too bright. It should be RGB = 6,6,6 – but that seems rather dark. RGB around 10,10,10 however looks fine.

A = Red channel of the regular normal map

Please note: As this texture is NOT a regular diffuse texture nor a normal map you have to switch its import settings to "Advanced" and check "ByPass sRGB Sampling" – or uncheck "sRGB (Color Texture)". Trilinear filtering is recommended.

AO (G) Translucency (b) Smoothness (A) contains the combined occlusion, translucency and smoothness texture. Its channels should be set up like this:

R = unused

G = Ambient Occlusion

B = Translucency

A = Smoothness

Please note: As this texture is NOT a regular diffuse texture nor a normal map you have to switch its import settings to "Advanced" and check "ByPass sRGB Sampling" – or uncheck "sRGB (Color Texture)".

Occlusion Strength Lets you control the strength of the final ambient occlusion which contains occlusion from texture input as well as from ambient occlusion baked into the vertex color (alpha).

Shadows

Enable extra Shadow Map Check this in case you want to use an extra texture for the shadow caster pass. [Find out more >](#)

Shadow Map Alpha (R) The texture that is used during the shadow caster pass if *Enable extra Shadow Map* is checked. This texture just contains the alpha or opacity and must store opacity in the red color channel. Use BC4 compression for best results.

Shadow Caster Culling In case you have pretty dense leaf meshes you may skip double sided rendering in the shadow caster pass (Off) and go with back face culling instead (Back). Doing so may only produce some barely noticeable visual artifacts but save some fill rate. [Setting culling to Back or Front needs deferred rendering.](#)

Lighting

Backface Normal Accuracy (BIRP only) The shader uses VFACE and thus may calculate proper normals when using single sided geometry. However this may produce rather harsh lighting revealing the single leaf planes. For this reason you may adjust the accuracy. 1.0 will give you proper lighting while 0 would results in normals equal to Speed Tree where back and front faces just share the same normal. In case you want to soften the lighting i recommend values between 0.35 and 0.45. [Such normals will always produce false specular highlights tho.](#)

Backface Smoothness lets you reduce smoothness on back faces (to fight false specular highlights or just because you want it). If set to 0.0 Unity fill fall back to pure lambert lighting – which usually you do not want.

Translucency Strength (BIRP and URP only) acts as a factor which gets multiplied with the translucency value sampled from the "AO (G) Translucency (b) Smoothness (A)" map and lets you fine adjust final translucency.

View Dependency (BIRP and URP only) 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.

Ambient Scattering (BIRP and URP only) The amount of scattered ambient lighting.

Fade out Translucency (BIRP only) Lets you fade out translucency (and smoothness) over distance as trees beyond the real time shadow distance tend to look a bit weird. If checked the leaf shader expects some global shader variables to be set as fade distance and fade range. You may set these manually from any script or use the provided `CTI.CTI_Utills.SetTranslucentLightingFade` function to so.

Wind

Wind Multipliers (XYZ) Multipliers for the baked bending strengths.

- **X** main bending
- **Y** secondary or branch bending
- **Z** edge flutter

Please note that these multipliers have to be synced across bark and leaf material as otherwise leaves will lose their connection.

Tumble Strength defines the strength of the tumbling animation.

Tumble Frequency lets you adjust the frequency of the tumbling.

Time Offset (BIRP only) lets you shift the tumble animation in time so it comes slightly after the main wind animation.

Enable Leaf Turbulence You have to check this to enable leaf turbulence.

Leaf Turbulence lets you adjust the strength of the turbulence.

Edge Flutter Influence lets you adjust the strength of the edge flutter (stored in vertex color green) affecting the leaf turbulence. Using edge flutter influence values above 0.0 will most likely add some distortion to the leaf meshes – which in fact looks really nice.

Enable normal rotation Checking this will make the vertex shader rotate the vertex normal according to tumbling. This is a bit more expensive but will improve lighting tremendously.

Leaf Noise (URP and HDRP only) Adds some noise to tumbling and turbulence based on the given phase and branch bending.

Use Wind from Script (BIRP only) In case you place your LOD trees using the terrain engine you have to check this in order to make the trees receive at least proper directional wind.

Fade out wind (BIRP only) In case your billboards do not use wind you may create an extra material just for the *last LOD* using mesh trees, assign it to the corresponding LOD and check this feature. Doing so will make the final transition between the mesh tree and the billboard a little bit smoother.

CTI/LOD Billboard shader

Shader inputs

Color Variation (RGB) Strength (A) Make sure that the color fits the one you have added to the mesh trees.

Albedo (RGB) Alpha/Occlusion (A) This slot should contain the provided albedo texture atlas.

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. A value of 0.45 should just be fine.

Alpha Leak Suppression: As the alpha channel of the Albedo textures stores both: *Alpha* and *Occlusion* dark pixels from the alpha mask might leak into the occlusion texture (caused by bilinear filtering) which would end up in full occlusion at the outer parts of the billboard. But if you set it about 0.6 all pixels darker than that will be set to white so you will get simply no occlusion on outer pixels – which in fact makes much more sense.

Normal (AG) Translucency (R) Smoothness (B) This slot should contain the created texture atlas.

Normal Scale Lets you scale the normal.

Specular Specular Color as a simple solid color which you most likely should set to the default value of dielectric materials which is RGB = 51,51,51.

Translucency Strength might be a bit different from the one set in the leaf shader as the billboard's translucency map contains some kind of depth map in order to reduce artifacts due to missing self shadowing.

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. The value here should match your settings in the LOD leaves shader.

Ambient Scatter The amount of scattered ambient lighting.

Tree Height Limit (legacy) lets you optimize fill rate in case the used billboard asset does not fit the actual shape of your tree.

Enable Wind lets you enable or disable wind animations on billboards.

Please note: Only Wind from script is supported.

Wind Strength As Billboards do not have any baked wind information you may use this parameter to make the bending of the billboard better match the bending of the mesh tree.