Contents:

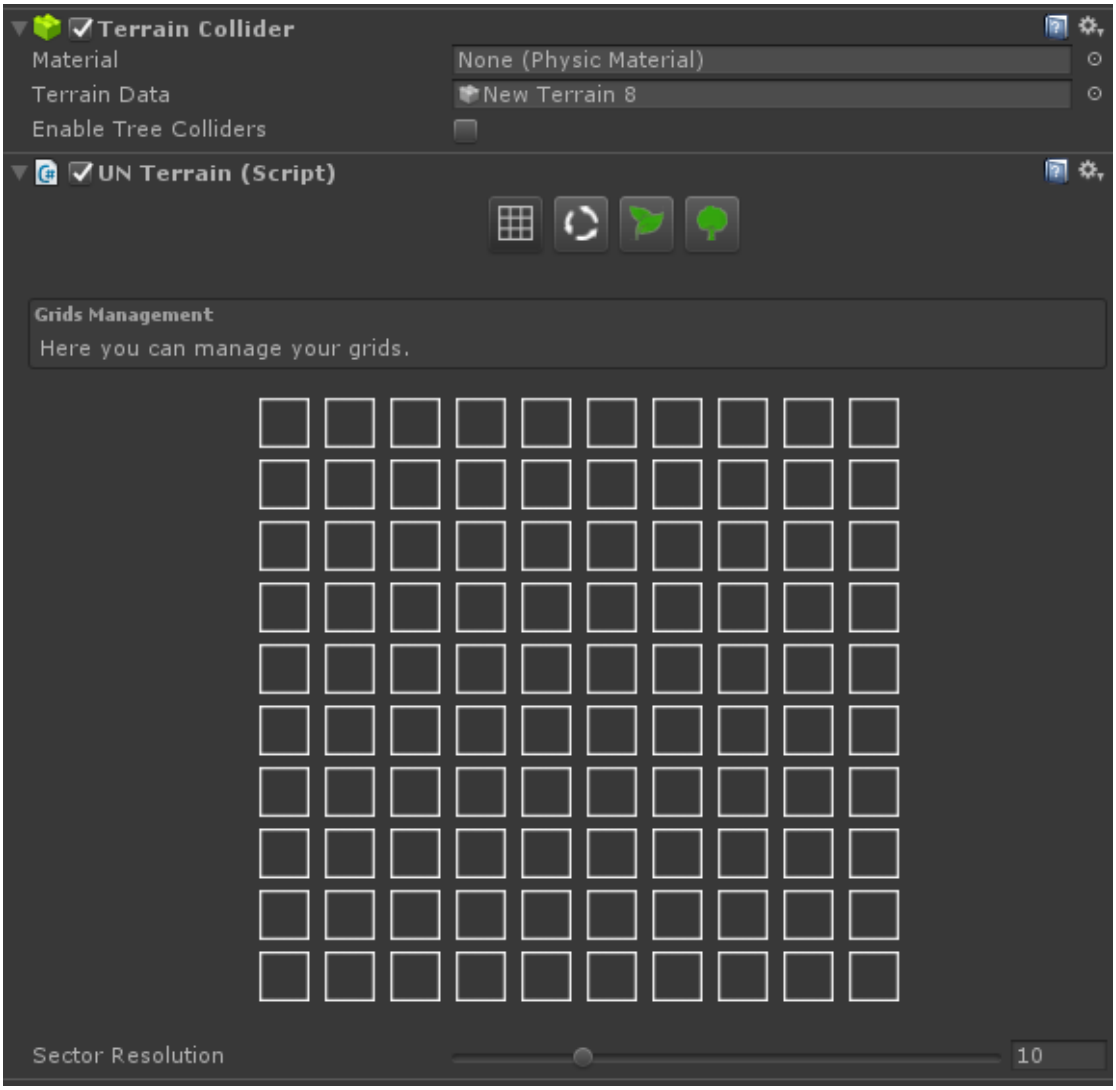
Trees

Getting Started:

This section will teach you how to integrate the trees mechanics into your own game easily without much effort.

First of all go to each of your terrains and add this component to them: "UNTerrain".

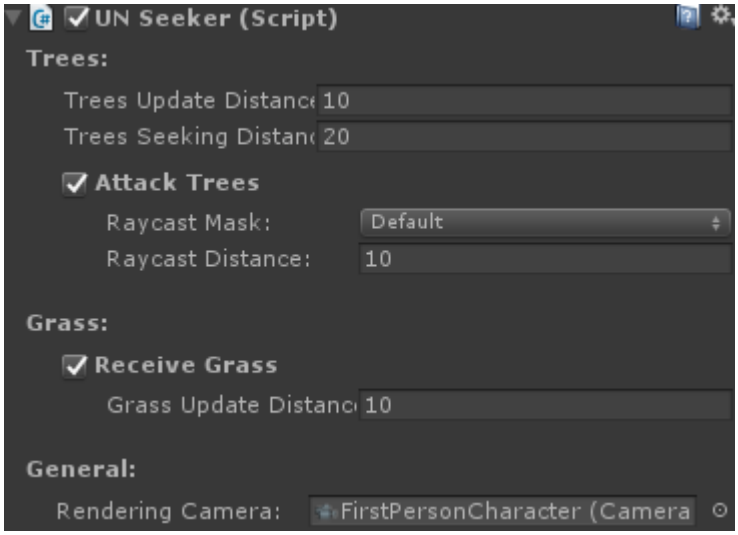
After you do that, disable the tree colliders and the terrain should look something like that:



And that's it, all of your terrains are set up and ready to use! Pretty easy isn't it?

(Note that if you want to add harvestable trees check out the pool items section)

But wait. Now you need to set up your player, so select it/them and add the "UNSeeker" component, it should look something like that:

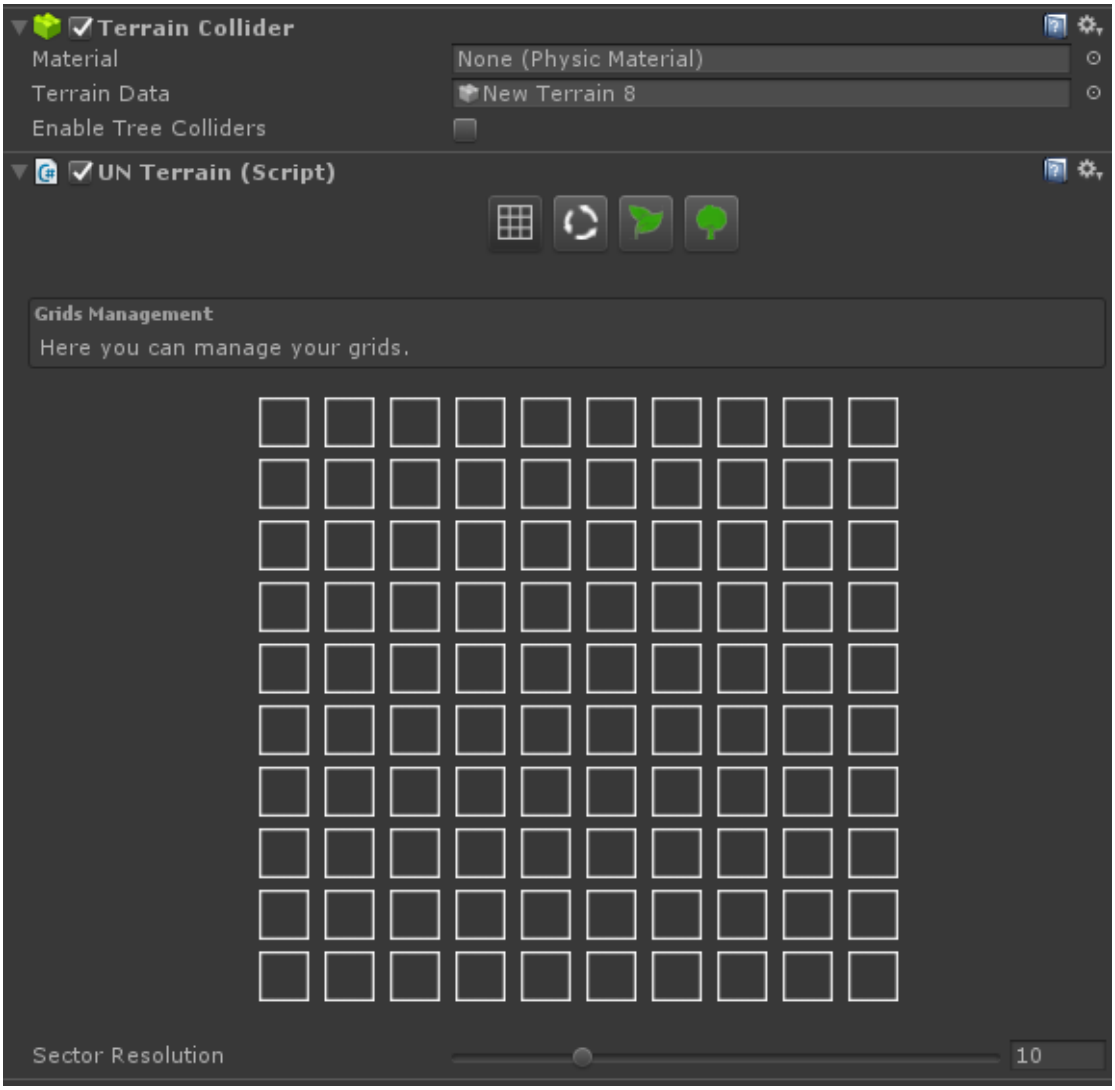


Make sure to put your camera on the "Player Camera" slot and you are ready to go!

UNTerrain:

The UNTerrain component is the component that handles all of the trees and also copying foliage from terrain.

Once you add the UNTerrain component to your terrain it should look like this:



As you can see there are 4 tabs that you can choose from:

Grid Management: This section will allow you to customize and preview your grids setup. You will need to find the right ratio for the resolution (for each 512 a 10 resolution would be good, so if your terrain size is 1024 put the resolution at 20 and so on [Note that it's not a must and won't make a huge difference])

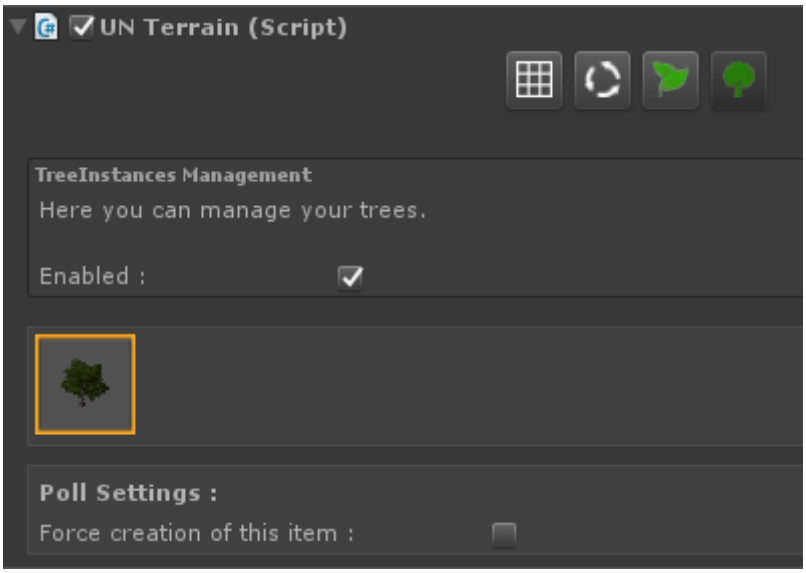
Pool Management: This section will allow you to customize your Pool settings. Where Pool Item Type is the type of Pool you will have (HarvestableTIPoolItem means harvestable trees)

And Pool amount means the amount of objects that will be generated for each used prototype times 2. (So 15 means 30 objects, 15 for colliders and 15 for game objects previews)

Grass Management: In this section you will find some tools that will help you copying your grass from this specific terrain etc. (Will be covered on the Foliage Chapter)

Trees Management: This section will show you the current trees you have on the terrain and will be updated every time you add/ remove a tree, or change the prefab of one of the trees on the terrain.

When selecting one of them you will have some settings that you can edit:



Force creation of this item: When the Pool is generated it checks if the specific tree is even placed on the terrain. If you plan on adding it while in runtime you need to make sure that you check this setting or it won't be interactable.

Pool Items:

Each one of the pool items needs to have a "Pool Item" which is a MonoBehaviour that has actions related to certain aspects.

For example:

1. HarvestableTIPoolItem – This is a pool item which is used on terrain tree instances, which makes them harvestable. (Network extensions use this component to sync the tree instance over the network. In case of changes, or custom harvesting, inherit from this class).

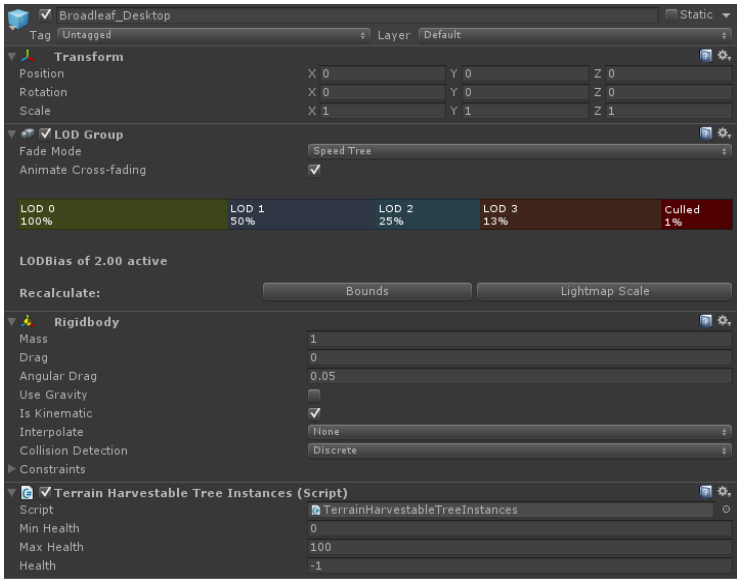
2. TerrainPoolItem – The basic pool item component which is used on terrain tree instances, simply making them intractable with no special additions. (Kind of like a "clean version").

Each tree instance automatically gets a pool item when created if they don’t already have one. (Automatically attached – this is explained on the "UNTerrain" section up above).

Some pool items also have variables which are quite important to edit if you like to get your own approach going, like min health and max health for example.

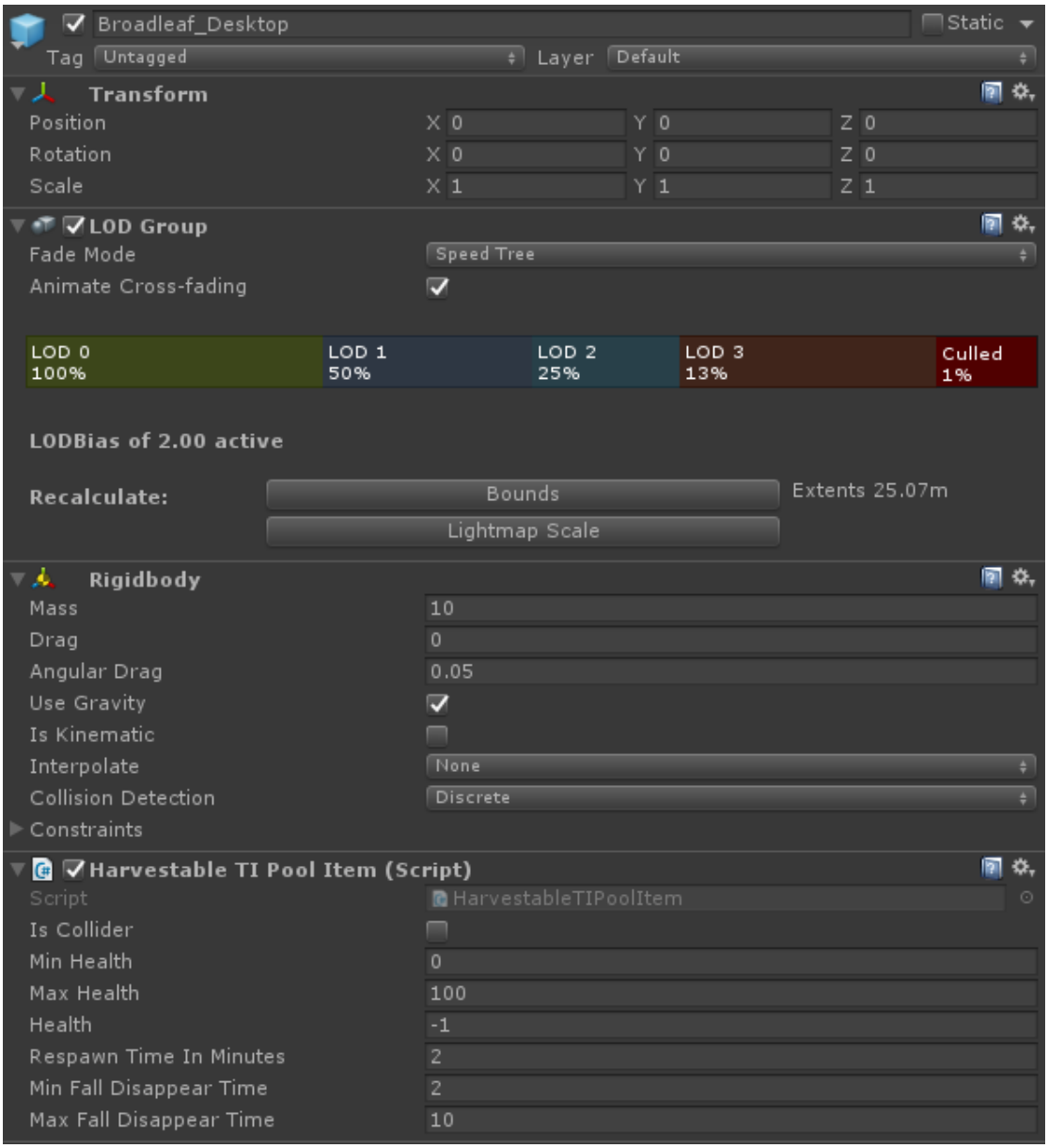
For that purpose so you don't need to edit the pool items every time you generate it (very frustrating), the system allows you to attach Pool Items on the prototype prefab before generating the pool.

For example:



As you can see the "Pool Item" is attached into this prefab, and all of the changes will be completely copied over to the both colliders, and tree instances that are generated by the pool system.

And that's pretty much it for the Pool Item, the rest is pretty much handled automagically by the system☺.



You can also learn how to customize and create your own Pool Items by looking into the API document and also checking out the Pool Items that comes with the system (for instance TerrainPoolItem).

Pool Component:

Pool component is a simple interface which can be attached to any MonoBehaviour and any Pool component which is assigned to the tree prototype will be replicated into the pool.

So why is that useful?

This allows you to have some custom callbacks and actions regarding the pool actions, such as when the item is Pooled, or returned to the Pool.

This allows you to get those callbacks without inheriting from a certain class, but you can use the pool component interface instead. (Also, note that only 1 pool item is allowed on an object, so that's also an advantage of the pool components).

In order to create a new pool component is as simple as that: (An example script)

```
using UnityEngine;

using System;
using System.Collections.Generic;

using uNature.Core.Pooling;

class PoolItemsNameAssigner : MonoBehaviour, IPoolComponent
{
    public string poolItemIdentificationName = "Pool Item";

    public void Awake()
    {
        Debug.Log("Pool item initialized : " +
poolItemIdentificationName);
    }
}
```

This class above will give you the ability to assign a custom name for each tree prototype and that way identify it in debugging, etc.

This exact script and exact variables will be "cloned" to the collider which will be generated for this tree prototype.

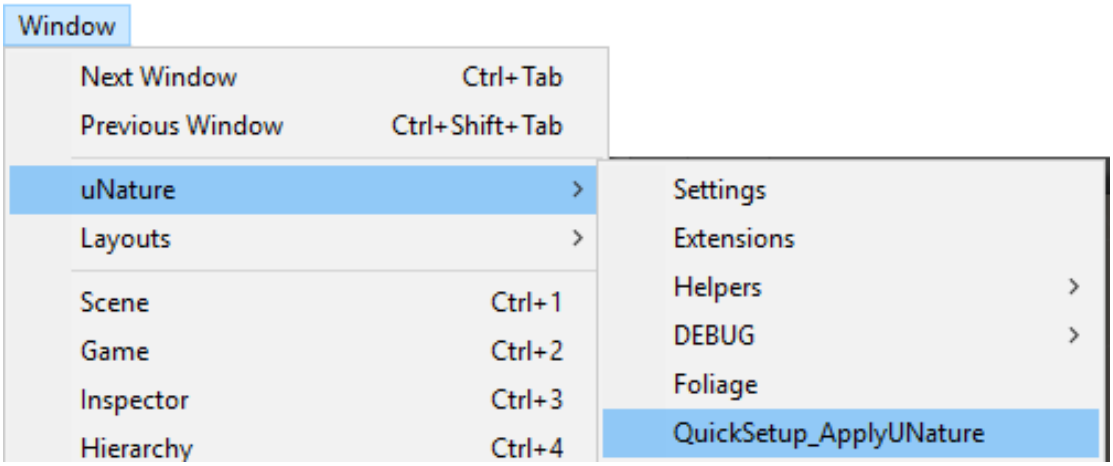
That's it for the pool components. They provide a way for you to attach a script to the pool without doing some hard coding into the pool item itself.

Foliage

Getting Started:

This section will show you how you can integrate the foliage mechanics into your game in a very short period of time.

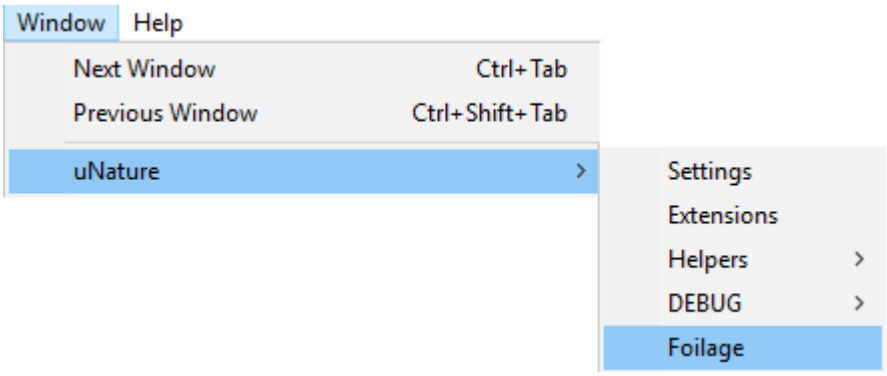
All you need to do is go to "Window/uNature/QuickSetup_ApplyUNature" which will quickly set up uNature in your scene.



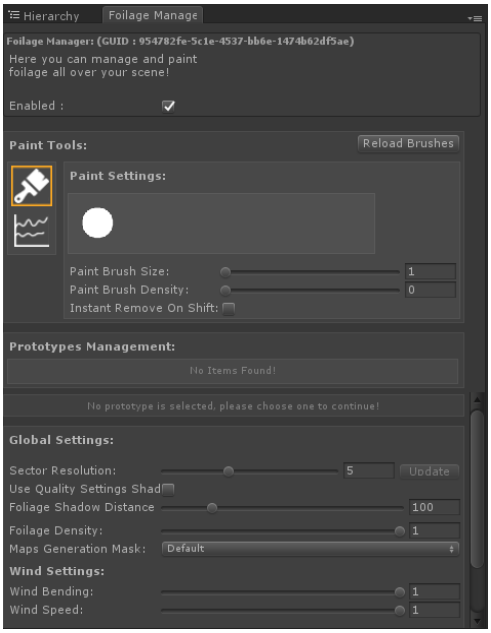
And that's it ☺

Foliage Manager:

This section will cover the foliage manager. As stated in the "Getting Started" section, you can open up the window by:



Again, I recommend putting it next to the inspector as such:



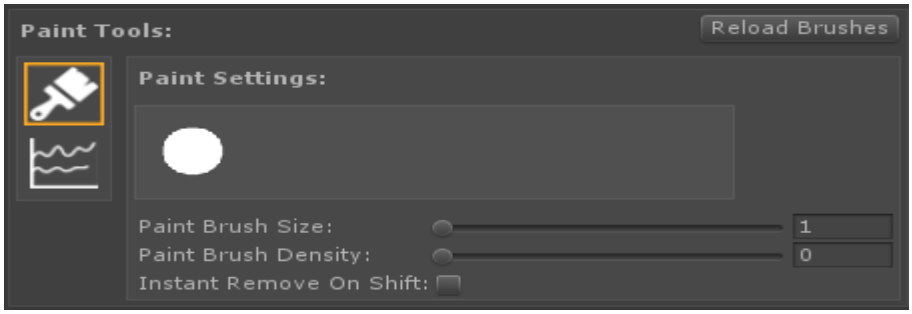
On the first glance it might look VERY complicated, but don't worry, it's not ☺

Just for the purpose of this tutorial, let's sort the foliage manager into 4 different sections:

1) Painting Section
2) Prototypes Section
3) Global-Settings Section
4) Per-Chunk Section.

Okay then, let's begin:

1) **Painting Section:**



In here you will see a few things, first of let's talk about the 2 different paint tools to the left.

This is the default brush which is just a normal brush as you would expect.

This one is a spline based brush which isn't working just yet. This will be expanded when it's added.

Let's now talk about the 3 different settings you have:

a) Paint Brush Size: the size of the brush, from 1 to 100.
b) Paint Brush Density: the density that will be drawn, from 0 to 15.
c) Instant Remove On Shift: When you want to decrease the density of an area you must hold shift and then it will decrease the density to the one you chose on the Paint Brush Density, although if you toggle this on it will automatically remove it to 0 when shift is held.

And lastly, the Reload Brushes button will re-fetch the brushes in the project, you shouldn’t have a use case for it but it will be covered in the "How to add custom brushes" section.

TIP: To stop painting without needing to disable the brush, click Escape on your keyboard!

2) Prototypes Section:

It looks complicated, but don’t worry, it's not as hard as it looks ☺

So let's start with the prototypes window:

This window will list all of your added prototypes. You can easily add a new prototype by dragging a texture/ a prefab to that window.

***Note: Meshes requires some changes to work properly, this will be covered down below.**

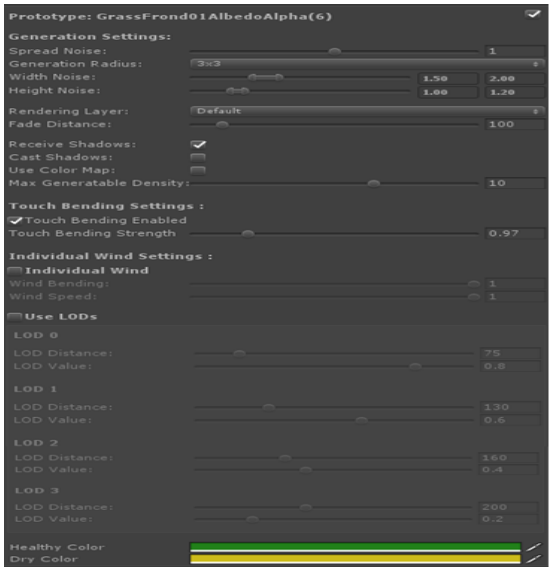
Once you have selected 1 or more prototypes you will see these 3 buttons popping up at the top right of the window:

"-": Remove the selected prototypes

"R": Reset the density of the selected prototypes from the scene.

"L": This will select the material instances of the selected prototypes. (This will be explained further down in the document)

Now if you select one of them the edit window under it should be enabled as well:



So let's start, first of all the toggle to the right of the prototype's name decides whether it will be drawn and calculated or not. Next are the variables:

Spread Noise: This is the spacing range of each density (from –value to +value)

Generation Radius: What amount of chunks will this prototype cover (3x3 means 9 chunks around the camera)

Width Noise: The noise range for the random width.

Height Noise: the noise range for the random height.

Rendering Layer: The layer on which the prototype will be rendered. Used to ignore reflections from water, and more

Fade Distance: The distance, on which the grass will start disappearing, note that increasing it might cause chunks to randomly popup, make sure to decrease sector resolution accordingly (will be explained on the global settings page)

Receive Shadows: Will this prototype receive shadows (doesn't affect performance)

Cast Shadows: Will this prototype cast shadows (significantly hurts performance)

Use Color Map: Will this prototype read the color from the surface and align to it

Max Generatable Density: The density which will be baked and calculated, the system will automatically decrease this for mesh grass to preserve performance. The lower it is, the less density that can be used and the higher the performance will be and the opposite.

Touch Bending Enabled: Whether touch bending will be calculated for this prototype (if disabled it can save performance)

Touch Bending Strength: The strength of the touch bending of this prototype.

Individual Wind: Whether this prototype will use custom wind settings (different ones from the global settings ones)

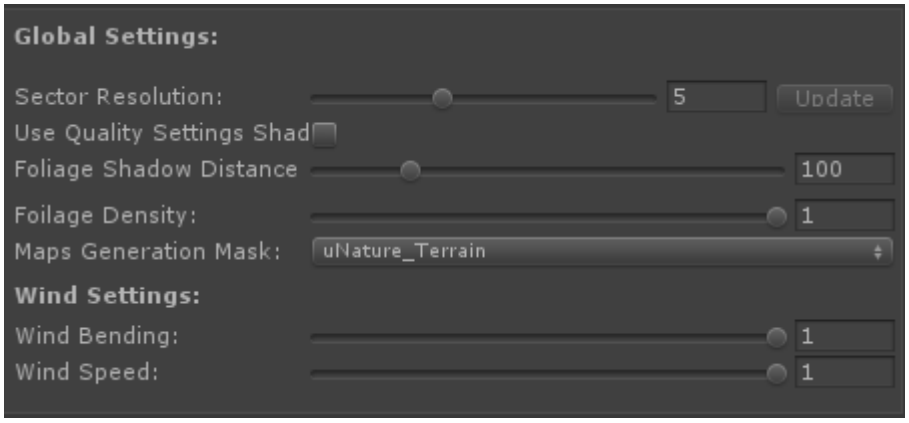
Wind Bending: The bending amount of the wind, the higher it is, the more the prototype will bend.

Wind Speed: The speed that the wind will be applied, the higher, the faster the wind will blow.

LODs: Will distance LODs will be applied, note that it doesn’t increase performance by a lot, but very helpful in very dense scenarios and doesn’t hurt performance, so it's always recommended to enable it.

Healthy Color & Dry Color: Using perlin noise, these 2 colors will be used to get a nice grouped color effect, similar to what unity's grass does.

3) Global Settings:



The global settings are different than the prototypes settings because they affect all of the prototypes. Well then, let's get started:

Sector Resolution: The resolution of the manager instances' sector (Will be explained in the Manager Instance section)

Use Quality Settings Shadow Distance: Do you wish to use the shadow distance from unity's quality settings

Foliage Shadow Distance: If you didn’t select the option above, select the shadow distance you wish to use for the foliage shadows.

Foliage Density: The global density multiplier for the foliage.

Maps Generation Mask: What layers will be taken into consideration when calculating the foliage heights. (Use this to ignore the player layer/ buildings layer so you won't end up having grass where you don’t want to have it)

Wind Bending: The bending of the prototypes' wind, note that this is only applied if the prototype doesn’t have custom wind settings enabled.

Wind Speed: The speed of the prototypes' wind, note that this is only applied if the prototype doesn’t have custom wind settings enabled.

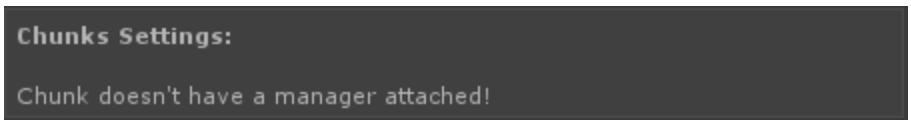
4) Per-Chunk Section:

This section allows you to edit each one of the manager instances (will be covered in the manager instances section).

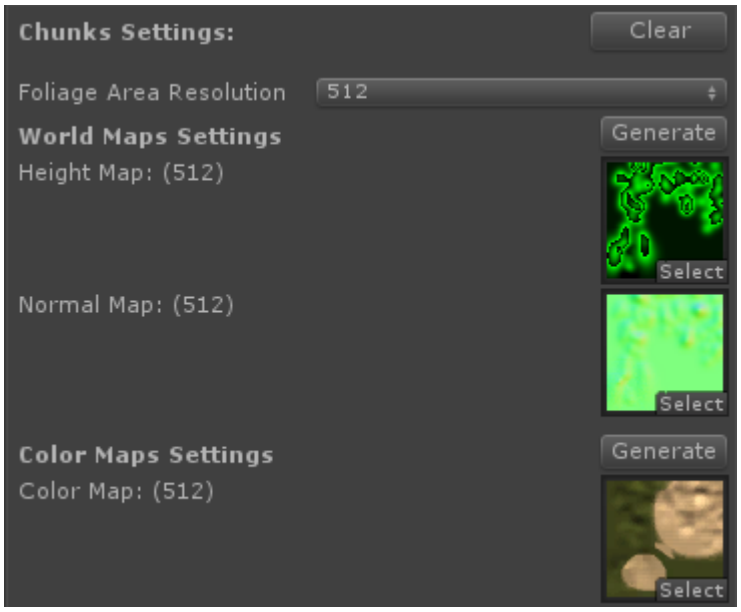
In short, the manager instances are 512x512 chunks on the world that are calculated procedurally every time you paint on a new area

So, in short, you want to simply left click on one of the black chunks on the scene and the window should change accordingly.

Now, if the "chunk" you clicked on is black, it means that there is no manager instance populated to it yet (means nothing is drawn/ was drawn on that area) and therefore it will show you that:



But, if it's pink, it means that something was/ is drawn on it and therefore you will get that result:



The "Clear" button at the top right will destroy that manager instance and clear all of the storage it takes from the project (world & color maps).

The Foliage Area Resolution is simply the resolution of that manager instance, please note that the size of each manager instance is 512, so the default resolution ratio is 1 (512/ 512 = 1)

Next up is the world map, the world map basically contains all of the world data regarding to that manager instance such as normals and heights. If you have changed the heights on that manager instance/ changed the mask layer on the global settings, make sure to hit "Generate" to apply height/normals changes.

And lastly, the color map is basically what it says, it's an texture representation of the mesh instance area, that way if you enable color maps on the prototype it can read the map colors.

Note that you need to regenerate the map if you change the textures on your terrain/ surface so the colors will be updated.

And that's it! This is basically the whole foliage manager window covered!

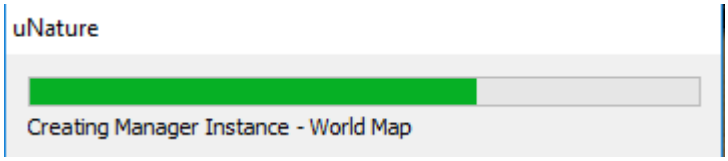
Procedural Manager Instances:

uNature allows you to draw all over on your scene, whether it's a rock, a tree, a human, a terrain or any kind of a surface as long as it has colliders, and well it doesn't matter if it's a unity collider, you can even add support for custom physics!

In order to allow that, uNature creates small chunks of 512x12 around the map, now, obviously they aren't all active at once which will cause a huge overload, but what happens is that the manager instances only get generated when you draw in their bounds.

The chunks are easily visible from the scene view as black squares that are scattered around the world. You might notice that some of them are pink (or none of them, it's completely fine) which means that they are populated and a manager instance has been created on that specific chunk.

When you paint on a non-populated chunk (which is black) you will get that progress bar:



This in short means that this specific manager instance is being generated and will soon be paintable and turn pink.

And that’s really all that you should know, unless you really want to go in-depth, which means digging into the code and you are more than welcome to do so! ;)

Mesh Prototypes:

If you remember in the Foliage Manager section it was written that adding a mesh prototype will require a bit more setup, well here it is:

Basically the way uNature works is that it does a lot of calculations on the GPU, that way you can save a lot of overload from the CPU.

Therefore uNature doesn’t work with every shader and you must change the shader to the uNature shader or make your shader compatible which will be explained in the "Supporting custom shaders" section.

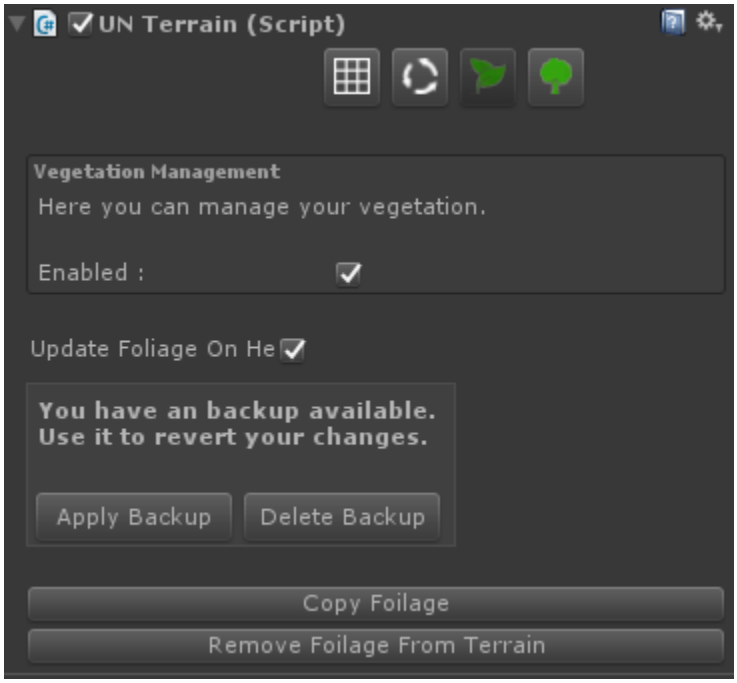
Now, if you already added the prototype and then you changed the material shader to uNature/FoliageShader you probably noticed that the grass still isn't drawn. That’s because when you add a prototype uNature creates a new custom material instance, so to edit it all you need to do is select the prototype/s on the foliage manager window and click on the "L" button which will select the material instances and then just change it to a compatible shader/ uNature's built-in shader (uNature/FoliageShader)

Copying Foliage From Terrain:

One of the key features of uNature is that it can copy the foliage from your terrain easily without you needing to do anything but clicking 1 button.

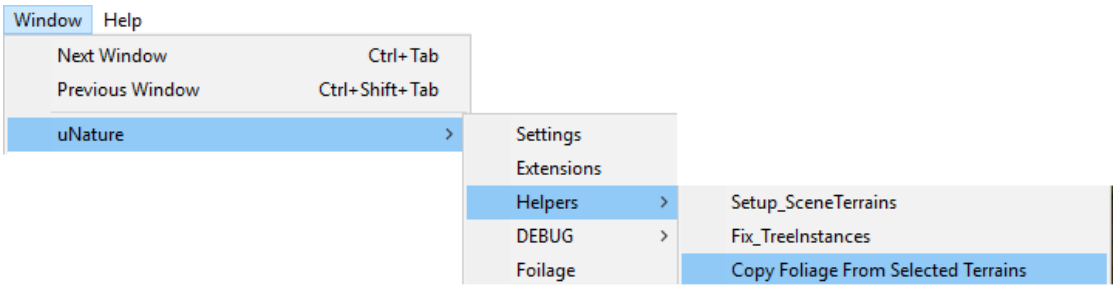
First of all, make sure you have a Foliage Manager created.

Now, There are currently 2 ways to do so, first one is adding an UNTerrain component to the terrain you want to copy and in the grass tab just hit "Copy Foliage"



The advantage of using this method is that it allows you to back up your terrain data so if it gets corrupted while copying the foliage you can always retrieve it to the way it was.

The other way is also very simple, select the terrains you want to copy on the inspector and simply click on "Window/uNature/Helpers/Copy Foliage From Selected Terrains":

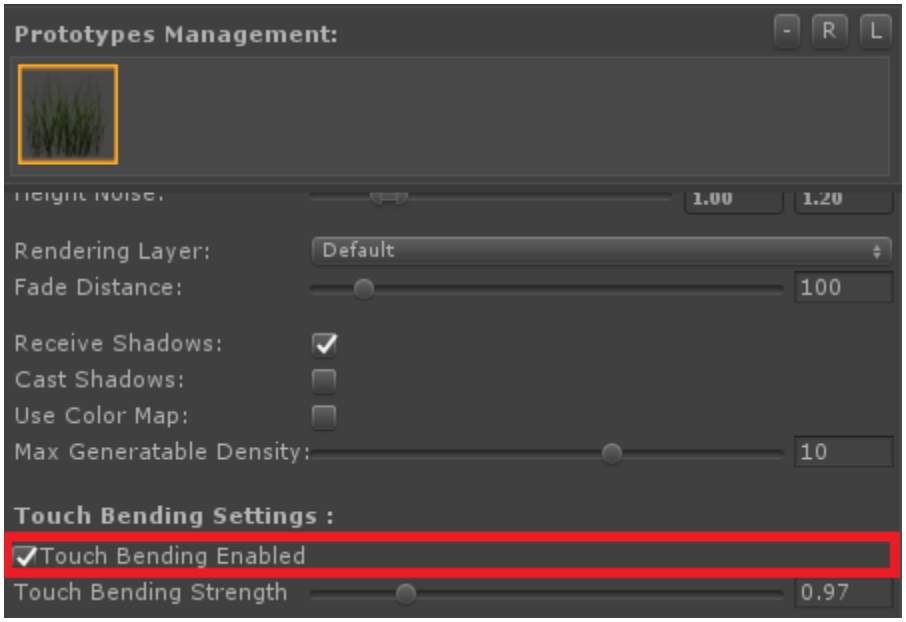


And that's it! ☺

Interactions:

uNature comes with a few out of the box interaction such as Touch Bending and Wind Source.

For the touch bending, all you need to do to get it to work is add the "Touch Bending" component to an object, and make sure that the prototypes that you want to have touch bending enabled have touch bending enabled.



Optimizing & Managing Performance:

uNature does a lot of optimizations for you out of the box, like decreasing the generated density based on each of the chunk's population, limiting density of a prototype by the amount of vertices it has and more.

But there's some stuff that uNature shouldn’t do manually, for instance, each of the manager instances has a sector resolution as described in the Foliage Manager section.

If you are using an insane amount of grass and want to reduce your view distance decreasing the fade distance on the prototypes alone won't be sufficient. What you want to do is also reduce the sector resolution accordingly.

One more thing that you can do is change the instance resolution from the foliage manager in the per-chunk which can help performance (reducing it).

One last thing that you can do is reducing the generate-able density on the prototype, reducing it will cause the system to show less density (so if you reduce it to 7 the max density that can be drawn for that prototype will be seven and so on), so if you are using only a density of 2 for instance, reducing it will help a lot. (Note that as described above, uNature already limits your max density automagically if your mesh is above a certain amount of vertices so reducing it more, will just help, but only if you really need to)

Overall, uNature does a lot of optimizations behind the scenes and you should be ready to go with the default setup, but if you still have performance issues, this will help ☺

Supporting Custom Brushes:

As you probably noticed, when you draw you can choose from several different brushes on the foliage manager and you can also add your own brushes if you wish!

All you have to do is use the psd template provided in the brushes folder and simply save it into the brushes folder as png.

Next, copy the settings from the brush that comes with uNature and you should be ready to go.

And that should be all ☺

(If the brushes aren’t loading in the foliage manager, just click the "Reload Brushes" button)

Supporting Custom Shaders:

uNature uses a lot of shader-side calculation to save overload from the CPU. What that means is that you need to modify your shader a bit in order for it to work with uNature.

It may sound complicated, but it's really easy and uNature already comes with tools to help you get it working.

So let's take this example shader for reference.

```
Shader "Example/Diffuse Simple" {
  SubShader {
    Tags { "RenderType" = "Opaque" }
    CGPROGRAM
    #pragma surface surf Lambert
    struct Input {
        float2 uv_MainTex;
        float4 color : COLOR;
    };
    void surf (Input IN, inout SurfaceOutput o) {
        fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * IN.color;

        o.Albedo = c.rgb;
        o.Alpha = c.a;

    }
    ENDCG
  }
  Fallback "Diffuse"
}
```

First of all we want to add vertex calculations to the shader, so change the pragma from this:

```
    #pragma surface surf Lambert
```

To:

```
  #pragma surface surf Lambert vertex:vert
```

Next, add the uNature helping tools to the shader:

Add `#include "uNature_Foliage_Base.cginc"` above the pragma that we changed a second ago.

And now you will also have to the vertex method to the actual shader and also add the gpu calculation tool from uNature, just like that:

```
Shader "Example/Diffuse Simple" {
  SubShader {
    Tags { "RenderType" = "Opaque" }
    CGPROGRAM
      #include "uNature_Foliage_Base.cginc"
    #pragma surface surf Lambert vertex:vert
```

```
      struct Input {
          float2 uv_MainTex;
          float4 color : COLOR;
      };

        void vert(inout uNature_Foliage_appdata v)
        {
                CalculateGPUVertex(v);
        }

      void surf (Input IN, inout SurfaceOutput o) {
          fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * IN.color;

          o.Albedo = c.rgb;
          o.Alpha = c.a;

      }
      ENDCG
    }
    Fallback "Diffuse"
  }
```

Next you must add some properties for uNature:

```
///UNATURE PROPERTIES BEGIN
                _Cutoff("Alpha cutoff", Range(0,1)) = 0.2
                _WindSpeed("Wind Speed", Range(0, 1.0)) = 0.2
                _WindBending("Wind Bending", Range(0, 1.0)) = 0.1
                _DensityMultiplier("Density Multiplier", Range(0, 1))
= 1
                _NoiseMultiplier("Noise Multiplier", Range(0, 2)) = 1

                [HideInInspector]
                _UseColorMap("Use Color Map", Range(0, 1)) = 0
                [HideInInspector]
                _ColorMap("Color Map", 2D) = "white" {}

                [HideInInspector]
                _GrassMap("Grass Map", 2D) = "white" {}
                [HideInInspector]
                _WorldMap("Foliage World Map", 2D) = "white" {}
                [HideInInspector]
                 InteractionMap("Foliage Interaction Map", 2D) =
"black" {}
                [HideInInspector]
                _InteractionMapRadius("Foliage Interaciton Radius",
Float) = 102

                _MinimumWidth("Min Width", Float) = 1.5
                _MaximumWidth("Max Width", Float) = 2

                _MinimumHeight("Min Height", Float) = 1
                _MaximumHeight("Max Height", Float) = 1.2

                lods_Enabled("LOD Enabled", Range(0, 1)) = 1

                lod0_Distance("LOD 0 Distance", Float) = 50
                lod0_Value("LOD 0 Distance", Range(0,1)) = 0.8

                lod1_Distance("LOD 1 Distance", Float) = 100
```

```
                lod1_Value("LOD 1 Distance", Range(0,1)) = 0.6

                lod2_Distance("LOD 2 Distance", Float) = 120
                lod2_Value("LOD 2 Distance", Range(0,1)) = 0.4

                lod3_Distance("LOD 3 Distance", Float) = 200
                lod3_Value("LOD 3 Distance", Range(0,1)) = 0.2

                [HideInInspector]
                _FoliageAreaSize("Grass Area Size", Float) = 1024
                [HideInInspector]
                _FoliageAreaResolution("Grass Area Resolution",
Float) = 2048
                [HideInInspector]
                _FoliageAreaPosition("Grass Area Position", Vector) =
(1,1,1,1)
                [HideInInspector]
                _FoliageWorldMapResolution("WorldMap Resolution",
Float) = 1024

                [HideInInspector]
                _FoliageInteractionPosition("Interaction Position",
Vector) = (0,0,0,0)

                [HideInInspector]
                _InteractionMapRadius("Interaction Map Radius",
Float) = 124

                _healthyColor("Healthy Color", Color) = (1,1,1,1)
                _dryColor("Dry Color", Color) = (1,1,1,1)

                fadeDistance("Fade Distance", Range(0, 10000)) = 100

                touchBendingEnabled("Touch Bending Enabled", Range(0,
1)) = 1
                touchBendingStrength("Touch Bending Strength",
Range(0, 10)) = 0.97
                ///UNATURE PROPERTIES END
```

All you need to do is copy paste it to the properties section, just like that:

```
Shader "Example/Diffuse Simple" {

    Properties
    {
                _MainTex("Base (RGB) Trans (A)", 2D) = "white" { }

                 ///UNATURE PROPERTIES BEGIN
                _Cutoff("Alpha cutoff", Range(0,1)) = 0.2
                _WindSpeed("Wind Speed", Range(0, 1.0)) = 0.2
                _WindBending("Wind Bending", Range(0, 1.0)) = 0.1
                _DensityMultiplier("Density Multiplier", Range(0, 1))
= 1
                _NoiseMultiplier("Noise Multiplier", Range(0, 2)) = 1

                [HideInInspector]
                _UseColorMap("Use Color Map", Range(0, 1)) = 0
                [HideInInspector]
                _ColorMap("Color Map", 2D) = "white" {}
```

```
                [HideInInspector]
                _GrassMap("Grass Map", 2D) = "white" {}
                [HideInInspector]
                _WorldMap("Foliage World Map", 2D) = "white" {}
                [HideInInspector]
                _InteractionMap("Foliage Interaction Map", 2D) =
"black" {}
                [HideInInspector]
                _InteractionMapRadius("Foliage Interaciton Radius",
Float) = 102

                _MinimumWidth("Min Width", Float) = 1.5
                _MaximumWidth("Max Width", Float) = 2

                _MinimumHeight("Min Height", Float) = 1
                _MaximumHeight("Max Height", Float) = 1.2

                lods_Enabled("LOD Enabled", Range(0, 1)) = 1

                lod0_Distance("LOD 0 Distance", Float) = 50
                lod0_Value("LOD 0 Distance", Range(0,1)) = 0.8

                lod1_Distance("LOD 1 Distance", Float) = 100
                lod1_Value("LOD 1 Distance", Range(0,1)) = 0.6

                lod2_Distance("LOD 2 Distance", Float) = 120
                lod2_Value("LOD 2 Distance", Range(0,1)) = 0.4

                lod3_Distance("LOD 3 Distance", Float) = 200
                lod3_Value("LOD 3 Distance", Range(0,1)) = 0.2

                [HideInInspector]
                _FoliageAreaSize("Grass Area Size", Float) = 1024
                [HideInInspector]
                _FoliageAreaResolution("Grass Area Resolution",
Float) = 2048
                [HideInInspector]
                _FoliageAreaPosition("Grass Area Position", Vector) =
(1,1,1,1)
                [HideInInspector]
                _FoliageWorldMapResolution("WorldMap Resolution",
Float) = 1024

                [HideInInspector]
                _FoliageInteractionPosition("Interaction Position",
Vector) = (0,0,0,0)

                [HideInInspector]
                _InteractionMapRadius("Interaction Map Radius",
Float) = 124

                _healthyColor("Healthy Color", Color) = (1,1,1,1)
                _dryColor("Dry Color", Color) = (1,1,1,1)

                fadeDistance("Fade Distance", Range(0, 10000)) = 100

                touchBendingEnabled("Touch Bending Enabled", Range(0,
1)) = 1
                touchBendingStrength("Touch Bending Strength",
Range(0, 10)) = 0.97
                ///UNATURE PROPERTIES END
```

```
    }

    SubShader {
      Tags { "RenderType" = "Opaque" }
      CGPROGRAM
      #include "uNature_Foliage_Base.cginc"
      #pragma surface surf Lambert vertex:vert
      struct Input {
          float2 uv_MainTex;
          float4 color : COLOR;
      };

        uniform sampler2D _MainTex;

        void vert(inout uNature_Foliage_appdata v)
        {
                CalculateGPUVertex(v);
        }

      void surf (Input IN, inout SurfaceOutput o) {
          fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * IN.color;

          o.Albedo = c.rgb;
          o.Alpha = c.a;
      }
      ENDCG
    }
    Fallback "Diffuse"
  }
```

And one last thing is just making the surface shader get cutoff:

```
      void surf (Input IN, inout SurfaceOutput o)
      {
          fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * IN.color;

          o.Albedo = c.rgb;
          o.Alpha = c.a;

          clip(o.Alpha - _Cutoff);
      }
```

Now, you might notice that you get an error about "uNature_Foliage_Base.cginc" not found. In order to fix this just move your shader to "uNature/Shaders/Foliage" and then compile it so it will apply the changes and that's it ☺

If you get into some issues while integrating your shader please email the support email and I will do my best to help ☺

Frequently Asked Questions:

Q: How do I stop grass from spawning on the player/ buildings and only spawn on areas that I want it to spawn on?

A: In uNature you have the ability to assign masks and then only the layers you want will be included, open up the foliage manager and choose the layers that grass will spawn on in the "Maps Generation Mask". Next, regenerate the world maps again on the corrupted chunks as explained in the Foliage Manager section.

Q: I changed the heights of the terrain and now the grass is sinking, how do I fix it?

A: Sometimes uNature will fail updating the grass heights automatically, in that case what you want to do is regenerate the world map on the corrupted chunks as explained in the Foliage Manager section.

Q: Grass shows up on the editor but not on the camera on runtime, what am I doing wrong?

A: Make sure that the camera has the "UNSeeker" component attached and that the "Is Grass Receiver" toggle is on.

Q: The foliage changes are saved on runtime/ aren't saved on runtime, how do I change it?

A: You can toggle runtime saving on/ off on the Settings window.

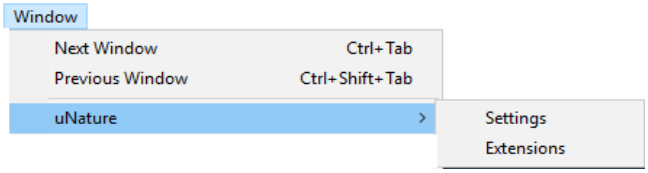
Editor Utility

Extensions:

The extensions system allows you to set up other systems to work with uNature quite easily without much effort.

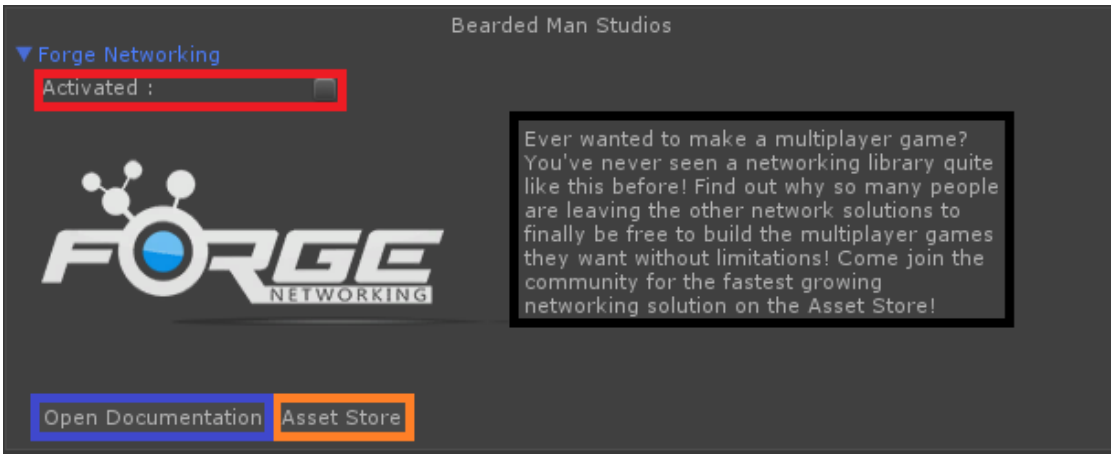
The system comes with several integrations as of now, such as Bolt Networking, Photon Cloud, UNET, UFPS, Forge networking and more.

In order to access the extensions window, do that:



And choose "Extensions".

This window will pop up:



You can choose what system you want to integrate, but in this case I opened up the "Forge Networking" Extension:

Activated: Is this asset integrated into the system ?

Description: The description of the asset.

Open Documentation: Open up the docs which will explain you how to integrate the system.

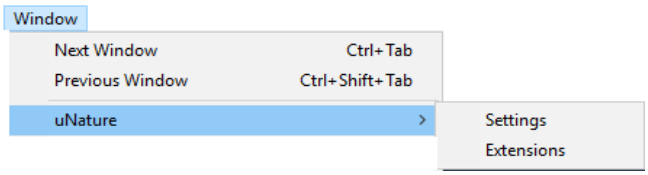
Asset Store: Open up the asset store page of the asset.

Some extensions have "Method Helpers" which are helpers buttons that help you set up the integration easily, the method helpers are explained in each one of the asset's documentation.

Settings:

The settings are global parameters that are used all over the features of the system.

Open up the settings window:



And this window will pop up:

Feel free to edit the settings and change it to work the way you want ☺.