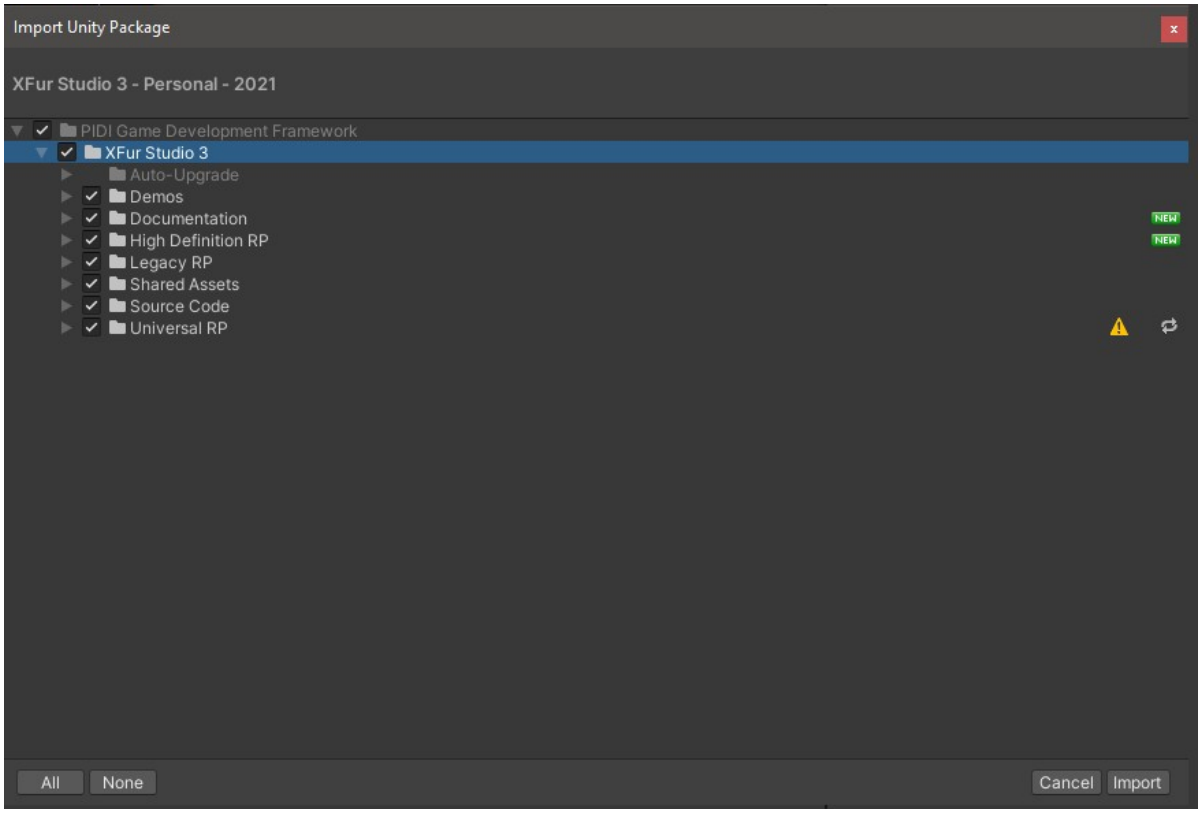
Thank you for buying XFur Studio™ 3. This offline version of the documentation is provided for your convenience, but you are **heavily encouraged to read the online version** instead as it contains the most up-to-date information about the product, its features and capabilities.

Installation

Installing the package is fairly easy. In the Unity Editor, open the Package Manager. Switch to “My Assets” and search for XFur Studio 3. Then press Download – > Import.

Import all the contents of the package and wait for them to be installed to your project. Depending on which pipeline you are using, you can deselect the folders of the other pipelines. For example, if you are using the Standard / Built-in Pipeline you can remove or skip importing the High Definition and Universal Rendering pipeline folders, etc..



To install either the Universal RP add on or the High Definition RP resources simply go to the folder with the name of the pipeline you are using within the PIDI – XFur Studio 3 folder.

Inside this folder you will find a **.unitypackage** with all the shaders, demos and content for the given pipeline. Unpack the .**unitypackage** file by double clicking on it and you will be ready to start using XFur Studio™ 3 on your SRP of choice.

Preparing your models for XFur Studio

There are a few requirements that all models intended for use with XFur Studio™ 3 have to follow. While models that do not follow these requirements may still work with our system it could cause different errors down the way, either with simple features such as Fur Grooming or advanced ones, like physics.

To ensure that your model will work in optimal conditions with XFur Studio™ 3 and XFur Studio™ – Core make sure that they:

- Have a uniform (1,1,1) scale when imported into Unity.
- Their local rotation & position must be 0,0,0 or as close to this value as possible.
- The red channel of the vertex colors of the model can be used to specify which areas must be covered in fur and which ones will not. In a future release, whether the vertex colors are read or not will be configurable, but at the moment it is always read. Most models, including third party ones, have a default value of 1 in the red channel, which means the whole model will be covered in fur at the start.
- The UV layout for your model should avoid, as much as possible, any overlapping islands. If your character has mirrored UVs (both sides of your mesh are overlapping to save texture space) this will break the grooming features, as the fur direction will not be properly stored. Physics / VFX and other effects that depend on precise mesh position / normals may also work incorrectly.
- A secondary UV layout can be used to control how the fur strands are distributed over the model's surface, but it is not necessary.

Many models provided in the Asset Store will, by default, fulfill all these requirements. If you are having problems, please check our troubleshooting guide at the end of this documentation or contact us with the details.

XFur Studio Database

XFur Studio™ works with a database that contains all the necessary materials for the different rendering pipelines to work. Global settings for those materials can be set up from the database as well. In this way, resources can be efficiently shared and reused across many different instances of XFur Studio™ 3.

Within the database's UI you can select the Rendering System to be used (whether it is Standard / Built-in, Universal or High Definition RP). The database will inform you which shaders have been loaded successfully, and whether they are found in your project or not. Any additional features (like the use of normal-maps in the Standard Pipeline or shadows in Universal RP) can be toggled globally from the database's UI.

While you can use directly the database provided with the asset, this is not recommended, and you should either duplicate it or create a brand new one for your project. In most cases only one database asset is necessary for the entire project.

Fur Strands Asset

The Strands asset provides the XFur Studio shaders with information about the shape and distribution of the strands that will form the fur of the character / model. These strands are stored in a texture's red and green channels, and for convenience, they can be procedurally generated through the Strands Asset itself or, if special shapes or additional customization is needed, a custom texture can be provided.

The default strands asset provided with XFur Studio™ 3 is procedural, and its first pass controls the red channel while the second pass controls the green channel of the strands texture. In the final XFur Studio shaders the red and green strands are used to provide color variation in the fur (the additional blue and alpha channels are reserved for future upgrades so in custom textures you should leave them empty).

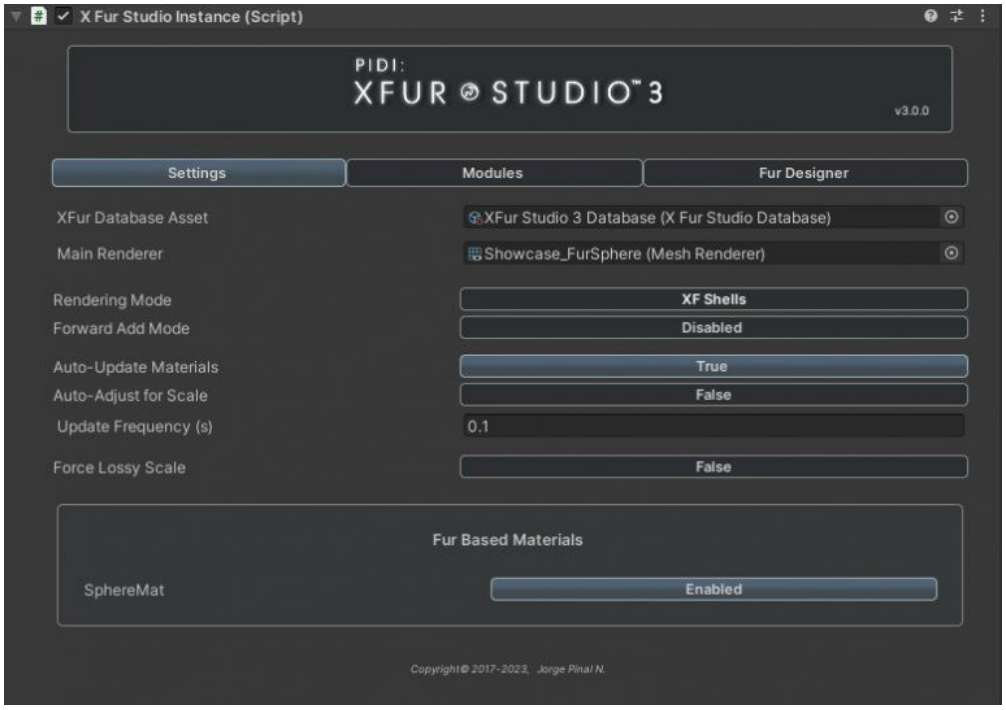
If you do not want any color variation in the strands you can use the parameters to generate only red or only green strands. Furthermore, this procedural texture can be exported for further editing.

XFur Studio Instance

In XFur Studio™ 3 the UI for our fur instances has been completely redesigned in order to make working with them more efficient. The basic UI presents three different buttons or tabs, each one with its own set of options. These tabs are the Settings, Modules and Fur Designer tabs.

Please notice that the XFur Studio™ Instance component should always be attached to the root object of whichever model you are using. This ensures that LODs can be used properly, and that the component will be able to keep track of all resources without issues.

Settings



In the Settings tab you can assign the XFur Database that will be used for this instance, as well as the main Renderer that will display the fur (this can be either a Mesh Renderer or a Skinned Mesh Renderer). For models using LOD, you should always assign the highest quality LOD to this slot.

Depending on the rendering pipeline you will see different settings, such as the Rendering Mode for the Built-in pipeline, which allows you to select between XF Shells (which make use of GPU Instancing and can be very efficient to render) or Basic Shells, which may be more lightweight on older devices.

You can enable the Forward Add mode, which is useful when using full lights and shadows in Forward Rendering mode (for Built-in, Deferred is recommended). The Auto-Update materials allows you to make changes to the fur properties at runtime and have them show up immediately, without having to call for a manual update. The Auto-adjust for scale feature modifies some properties of the fur so that they remain consistent with models that are either bigger or smaller than the uniform scale (1,1,1). The Force Lossy Scale feature can be used to correct some issues present in models that, from the source, do not have an applied uniform scale.

Finally, a series of toggles are displayed in the Fur Based Materials box, one for each material in the renderer. These toggles allow you to enable or disable fur rendering from each one of the materials, giving you great control over the appearance of your model.

Modules

The modules tab allows you to enable or disable each one of the advanced XFur Studio™ modules, including the physics, LOD and VFX modules. Each module has its own set of options that will be displayed in this section as you enable them. We will cover in more detail what each module does and how to set them up, in their own sections of this documentation.

Fur Designer

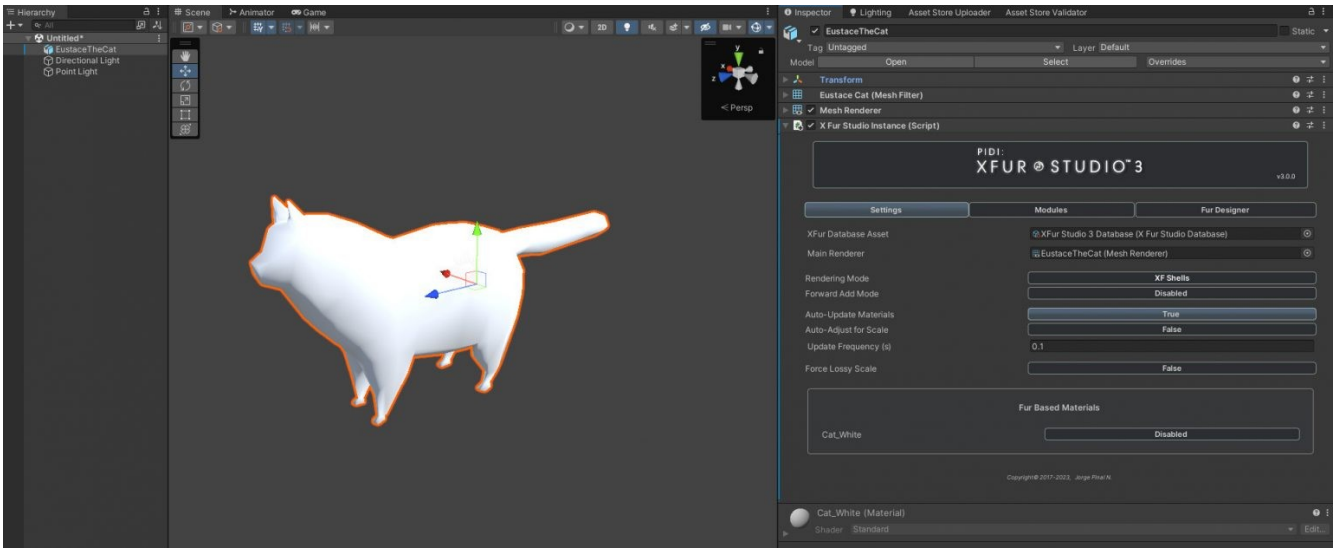
In XFur Studio™ 2 all fur properties for each fur material were displayed as a series of drop down menus that linked together and made the UI difficult to navigate for some users. Further adjustments, such as grooming and the creation of Fur Data maps had to be done in a completely separated scene using an additional piece of in-editor painting software called XFur Designer.

This workflow could easily turn cumbersome. To solve these issues, all fur editing is now done through the brand new XFur Studio Designer window, a completely in-Editor suite of tools and panels that can be opened from any XFur Studio Instance by just selecting the Fur Material you wish to edit and pressing “Enter Edit Mode”.

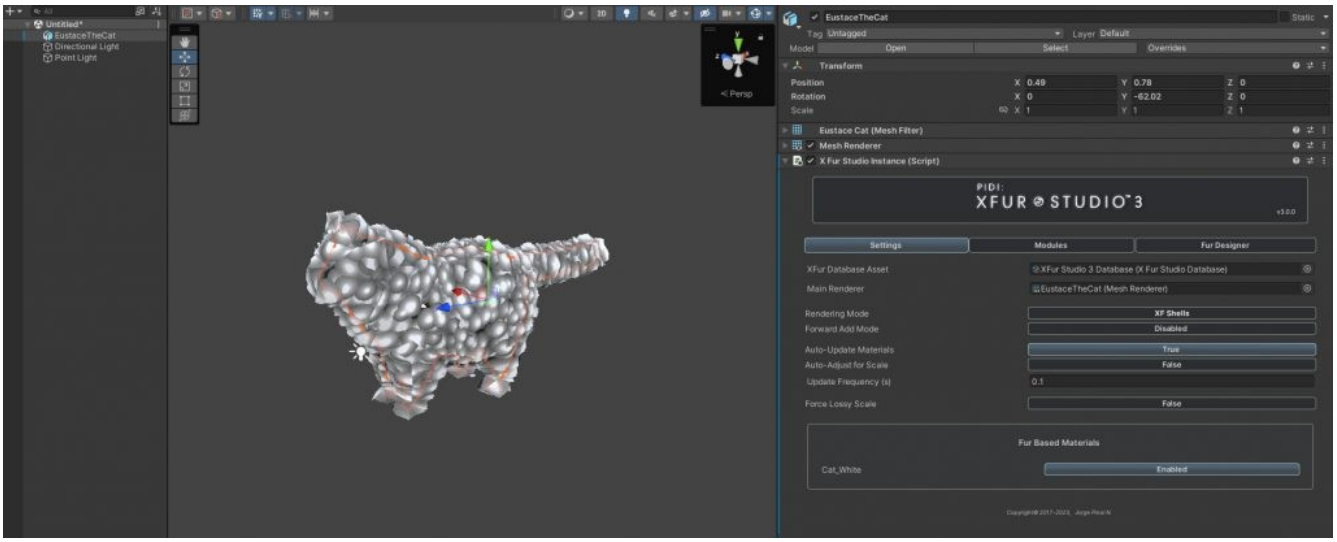
Once in Edit mode, the brand new XFur Studio Designer window will be displayed. There are several tools available in this window, including a Main Settings panel, a Properties panel and several different 3D painting brushes that will allow you to finely tune the appearance of the fur that covers your model by creating Fur Data Maps. All these settings can be exported to re-usable Fur Profile assets so that they can be applied to other models, by loading them through the same UI. XFur Studio Designer is a complex and fully featured set of tools, and as such it will be covered in detail in the next section of this documentation.

Fur Settings

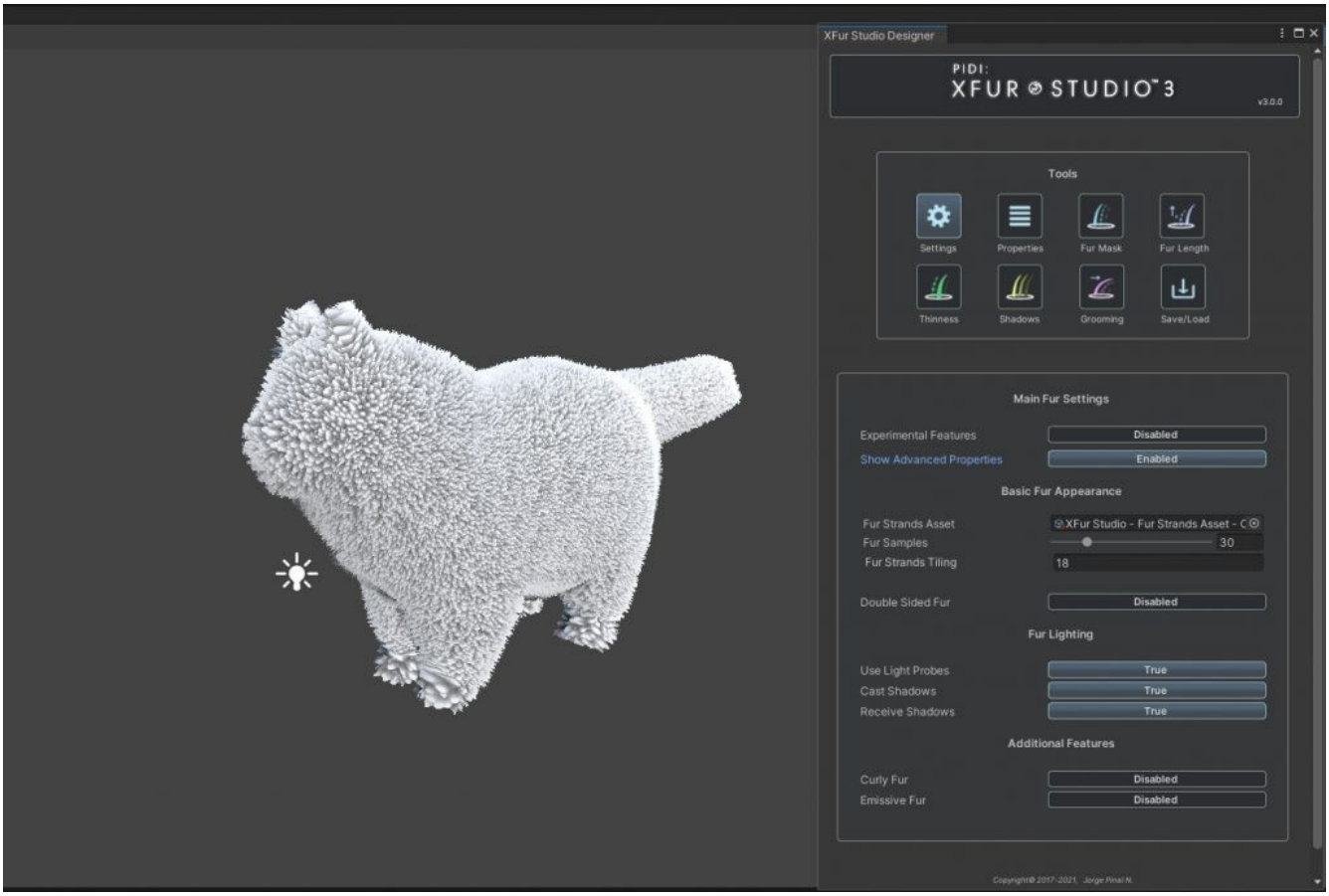
In order to understand all the features available in XFur Studio™ Designer, we have approached this documentation as a simple tutorial that will ask you to work on the “Eustace the Cat” model, included with the asset. To begin with this, let’s add Eustace to any scene, select its root object and add the XFur Studio Instance component to it. Then, press the Settings tab.



At this point the component will have already attempted to automatically assign as many of the needed resources as possible, including the first XFur Database and the first XFur Strands Asset that it could find, as well as the first renderer available in your model. If everything seems correct, enable fur rendering in the Cat_White material by changing its value to “Enabled”. Fur will be displayed immediately. However, this fur will most likely not look completely right.



To solve this, we will switch to the Fur Designer tab and press Enter Edit Mode. This will open the XFur Studio Designer window, in its Settings panel.



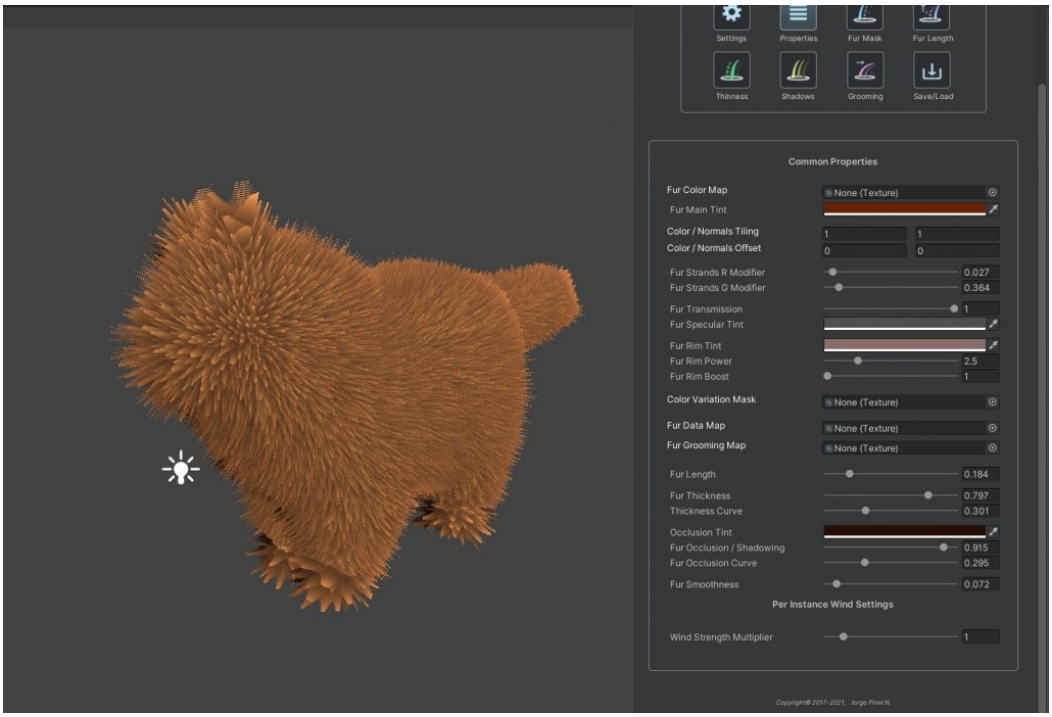
In this panel you can enable Experimental and Advanced features for the fur rendering, such as Rim lighting and strand color variation settings. You can select which Fur Strands asset to use in order to control the appearance of the fur, the amount of fur samples to render (more samples produce a better-looking fur at the cost of performance) as well as the tiling of the fur strands map. For this tutorial, we are going with a tiling of 18, using the Curly Procedural strands and enabling advanced properties. For desktop / consoles any number of samples between 16 and 30 will produce great results. For mobile and lower end devices aim for a sample count under 16 fur samples.

Additional settings available to control are whether the fur will interact with light probes as well as whether it will receive / cast shadows or not. In the Universal RP the shadow casting properties have to be defined from the XFur Studio Database since it depends on additional shader-based keywords.

Curly fur properties as well as the emissive channel (useful for fantasy creatures) can also be enabled from this panel. Once you have your initial setup ready, we can move on to the Properties panel.

Fur Properties

In the properties tab of the XFur Designer window you will find most of the configurable settings for the fur of your characters.



Fur Color Map is the texture map that contains all the color information for the fur. It is equivalent to the Albedo map in the Standard shader. The Fur Main Tint is the overall tint that will be applied to the fur covering the whole model.

The **Color/Normals** tiling and offset controls how the Fur Color map (and if available, the Normal map) should be tiled and offset at a UV level. The **Fur Strands R & G Modifiers** control the slight color variations between the R and G channels of the Fur Strands Assets. These values can be helpful to give a more natural color to the fur. The **Fur Transmission** value controls (in Universal RP and HDRP) how much light can go through the fur.

The **Fur Specular Tint, Rim Tint, Power and Boost** are self explanatory and control the appearance and intensity of the specularity and rim lighting applied to the fur.

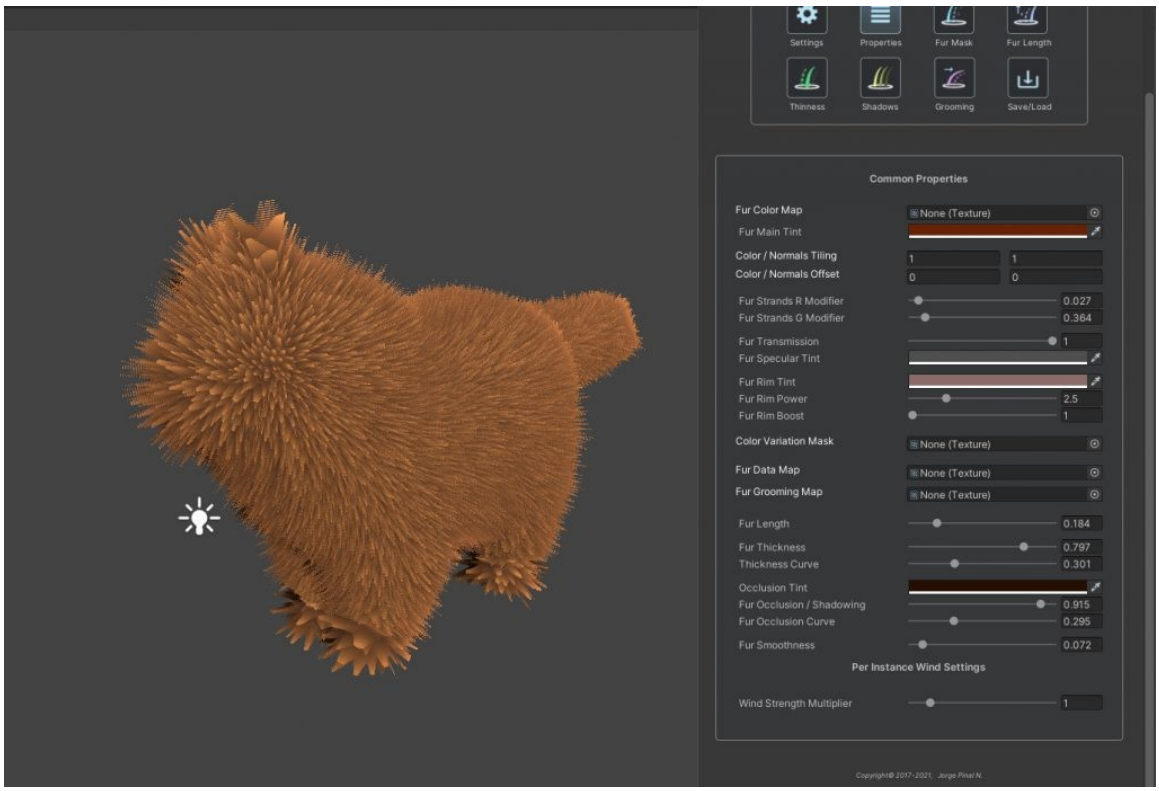
The **Color Variation Mask** allows you to apply four different tints to the fur using this texture as a Splat Map, with each channel representing a different tint.

The **Fur Data** map controls which areas of the model are covered with fur (Red channel), its length (green channel), thickness (alpha channel) and shadowing (blue channel).

The **Fur Grooming** map controls the direction in which the fur grows, storing tangent-based vectors.

The next series of properties control the **Fur's Length, Thickness**, a **Thickness Curve** going from root to tip, **Occlusion** and **Occlusion Tint**, as well as an **Occlusion Curve**. Finally there are two sliders to control the **Smoothness** of the fur and the intensity of the Wind applied through the **XFur Studio Wind Zone** object.

For the image above, we have reduced the thickness of the fur to produce thinner strands. We also modified the Occlusion to reduce its strength and give the character a fluffier appearance.



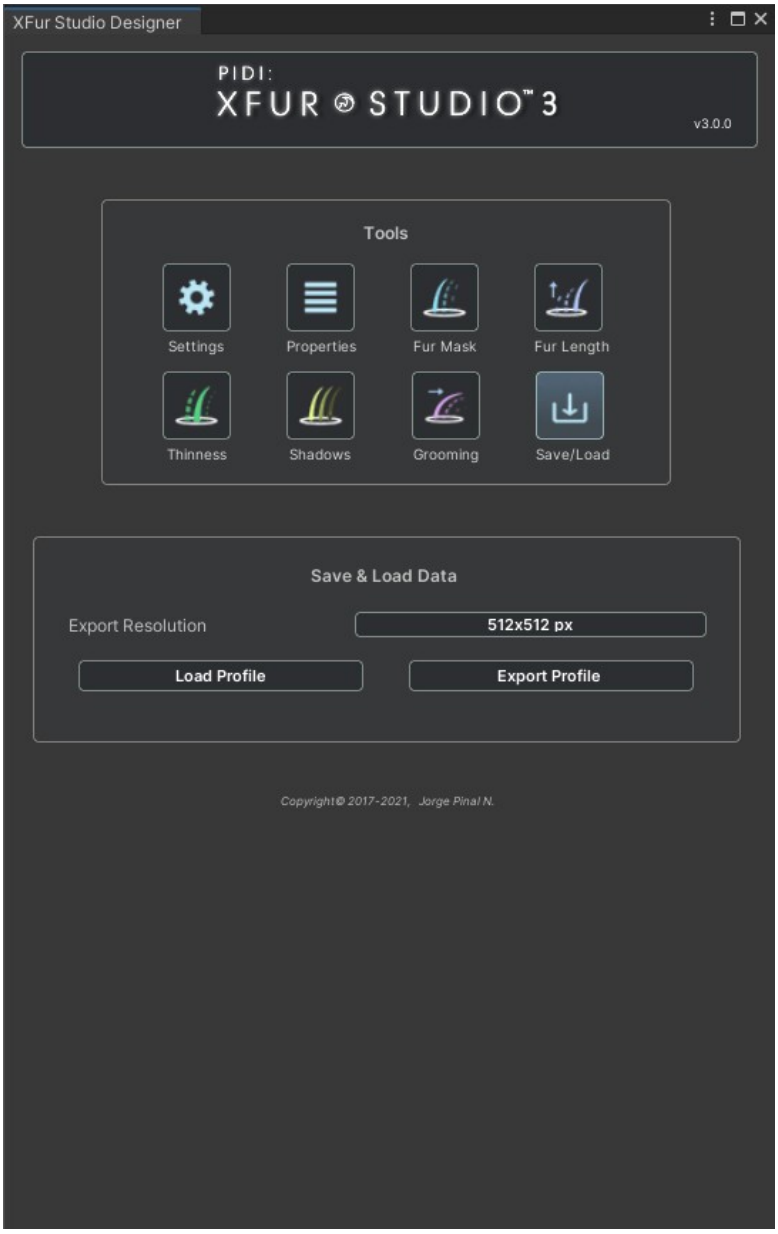
Once you are happy with the way the fur looks once you have modified all of its settings and painted its properties, then you can head to the Save/Load tab.

Below you can find a small video timelapse showcasing how fur was customized and adjusted for one of our models, a German Shepherd Dog.

https://youtu.be/WlcFJ4JjB5M

Save & Load Profiles

At any moment when editing a model using XFur Studio, you can switch to the Save/Load tab where you will find a set of options that will allow you to save your work to an external Fur Profile asset or, alternatively, load a previously saved Profile to modify the fur's appearance. Doing this is fairly easy, you just need to set up the export resolution when exporting the fur profile and click export, or alternatively press the Load Profile button and select the file to load.

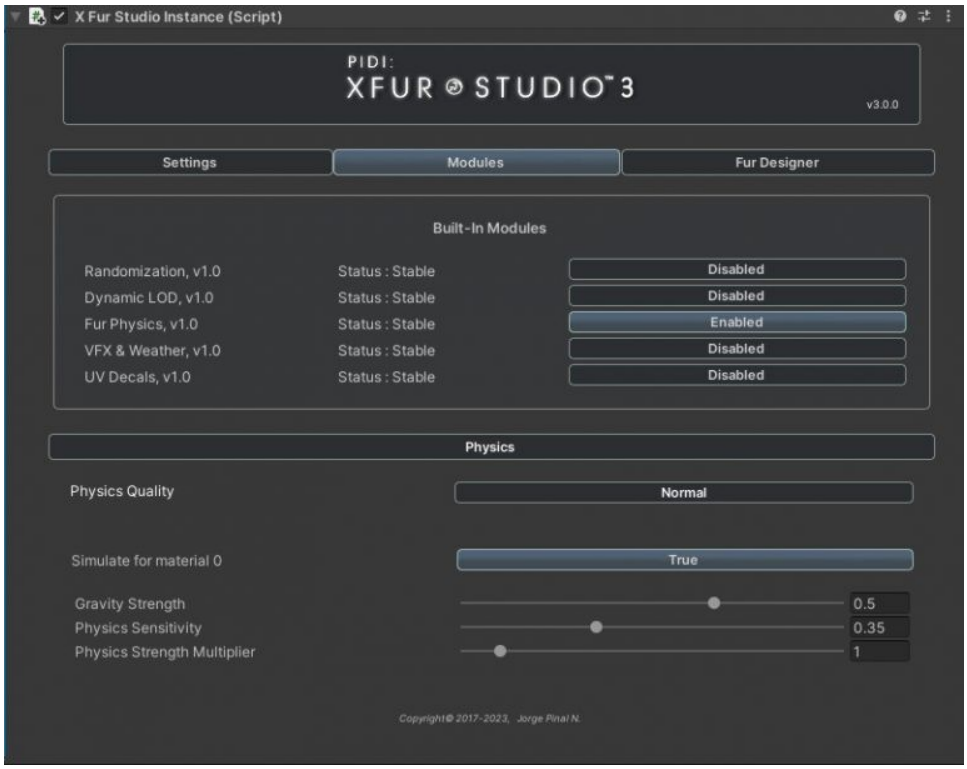


Advanced Topics

All the advanced topics in this section of the documentation cover features not available in XFur Studio™ Core.

Physics Module

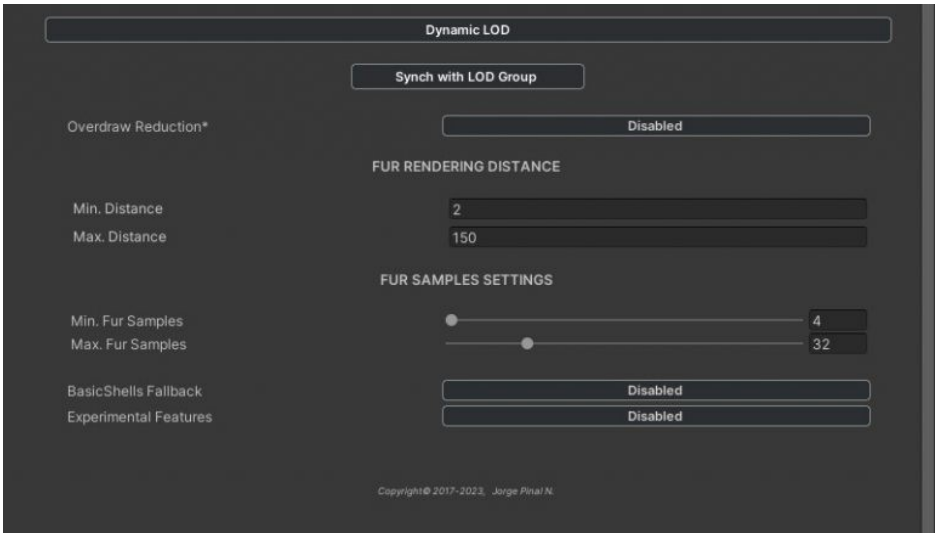
The physics module allows you to simulate basic physics on the fur of your characters. It is entirely run on the GPU so its performance impact is extremely low, and its memory footprint can be adjusted by changing the quality of the Physics Simulation. Low and Very Low qualities will be enough for most models and in its lowest setting the Physics module uses less than 10Kb of memory for each material using the physics simulation. To use it, simply enable the module in the General Settings tab and then enable the simulation for the corresponding fur materials in the Physics Module tab.



The Gravity strength and the Physics Sensitivity settings are shared across all materials in the instance and while at release fur length has the most influence over the physics effect, an additional Physics Strength Multiplier variable is provided to further adjust how strong the final effect is.

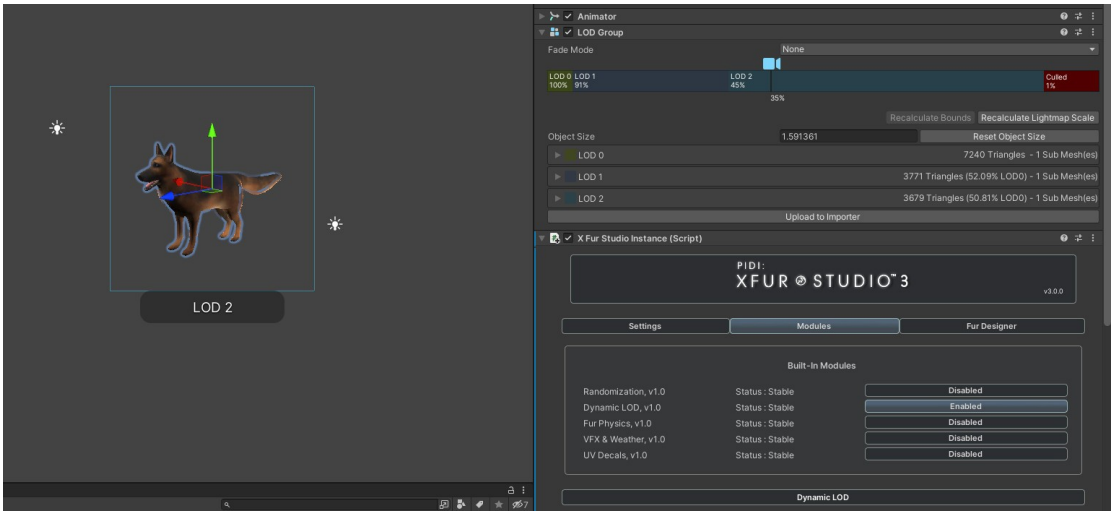
LOD Module

The Dynamic LOD module allows you to dynamically adjust the quality and features of XFur depending on the distance between the character and the camera. It is compatible with the standard Unity LOD Group component, adding fur to each active LOD level when needed and aiding you to keep a stable performance.



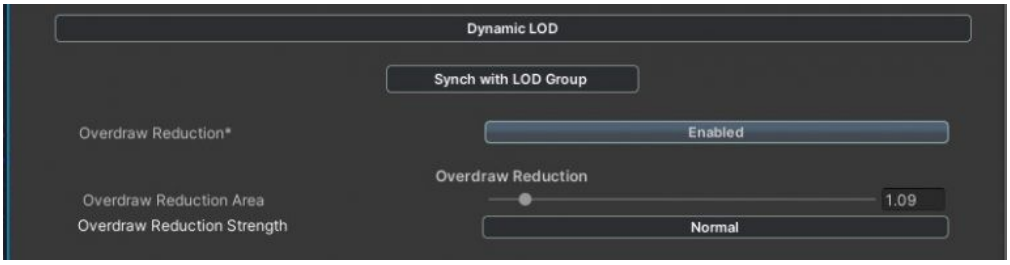
The component will ensure that the amount of fur samples used by XFur will move from Min. Fur Samples to Max. Fur Samples as the character gets far from the camera, reaching Max. Fur Samples when it is a Min. Distance and Min. Fur Samples when it is at Max. Distance. If Basic Shells is enabled, once the character is further away than the Switch At Distance value it will automatically switch to the Basic Shells rendering mode.

Pressing the Synch with LOD Group will automatically find the LOD Group component attached next to the XFur Studio Instance component, read the first Renderer assigned to each LOD Level and automatically add fur to it when needed.



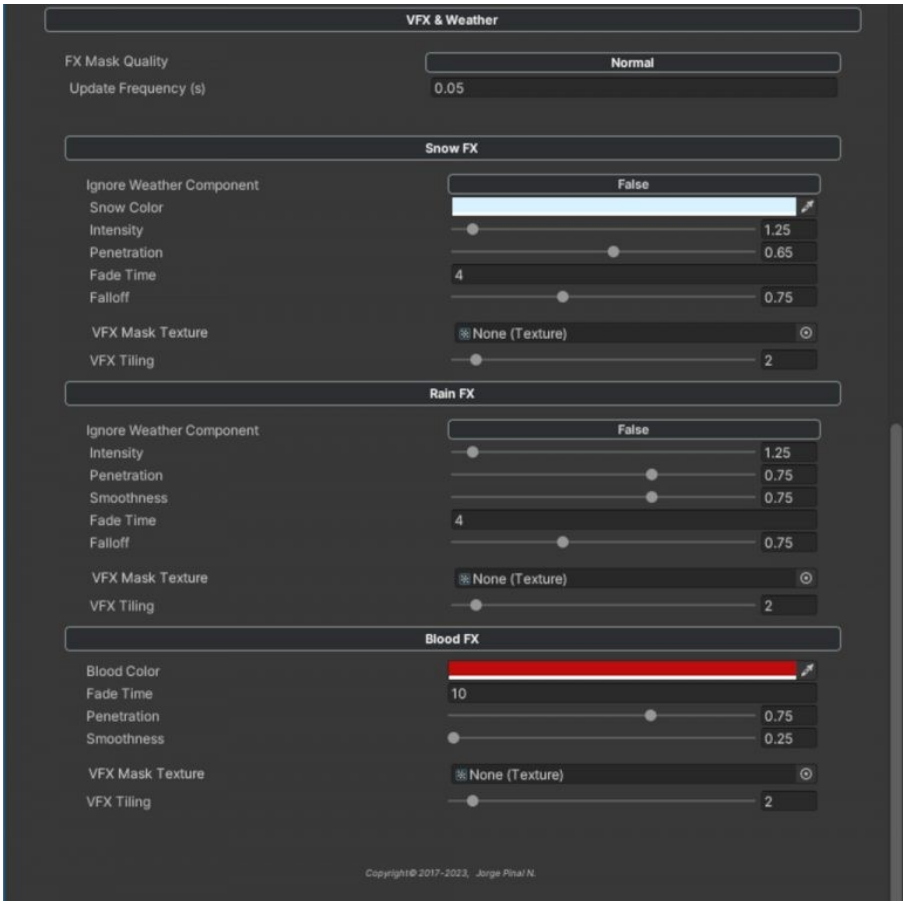
NOTICE: The Synch with LOD Group feature works only on the first renderer attached to each LOD Level. If your character is made of multiple renderers or the mesh using fur is NOT the first renderer assigned to each LOD Level, you should not use the Synch with LOD Group button. Instead, you may enable the experimental features in the LOD module and manually set up each LOD Level renderer. Be warned that Experimental features may cause unexpected errors.

Another feature available with the Dynamic LOD module is Overdraw reduction. IT allows you to reduce the amount of passes of fur directly in front of the camera, where they are less noticeable, in order to improve performance slightly.



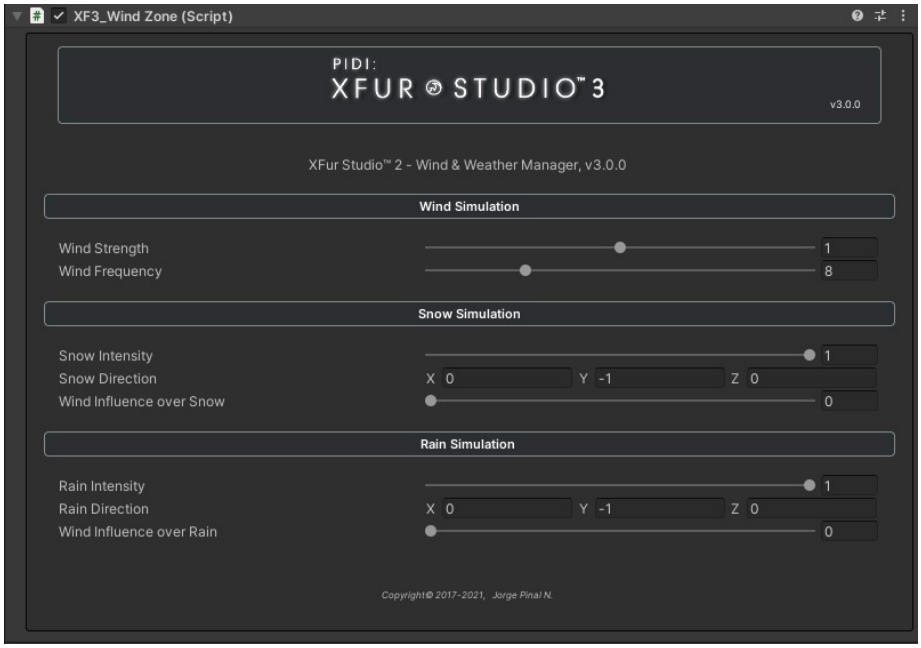
VFX Module

The VFX module lets you simulate snow coverage, blood effects and rain on the fur. These effects can be adjusted on a per-instance basis but can also be influenced by the XFur Studio 3 Wind Zone object in the form of global rain and snow effects with unique directions and intensities.



For each effect you can adjust the intensity of its progressive coverage (in the case of rain and snow), the speed with which they fade away in seconds, their falloff over the normals of the model (again for the snow and rain) as well as an additional VFX mask with tiling to provide even more variety to the way the effect is projected over the character's fur. To disable either the Snow or the Rain effects, simply set their intensity to 0. The blood effect is applied on demand, so unless a Painter object applying the Blood effect interacts with the character it will not be visible.

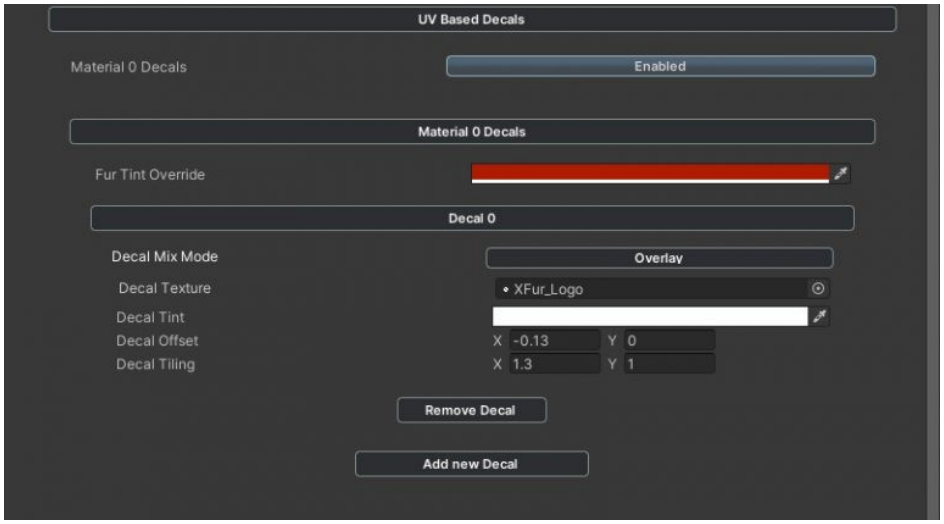
Finally, the XFurStudio3_VFXModule is deeply linked to the XFurStudio3_WindZone object, which handles the global intensity of the wind, snow and rain simulations as well as their global directions. To add a WindZone to the scene, create an empty object and add the XFurStudio3_WindZone component to it.



The Snow intensity and Rain intensity values provided by the WindZone script are further multiplied by the Snow and Rain intensity values in the VFX Module of each instance. This means that setting the Rain intensity in the Wind Zone object to 0 will stop the rain simulation across all instances while setting it to 0 in the VFX module will disable it only for that particular instance. When no Wind Zone object is available, Snow and Rain intensities are set to 1 by default and their direction is set fully downwards (0, -1, 0)

The wind simulation provided by the Wind Zone script also influences the direction of the Snow and Rain effects for a more realistic behavior. You can control the influence of the wind strength (and its direction) over the Snow and Rain effects independently on their corresponding tabs

UV Based Decals



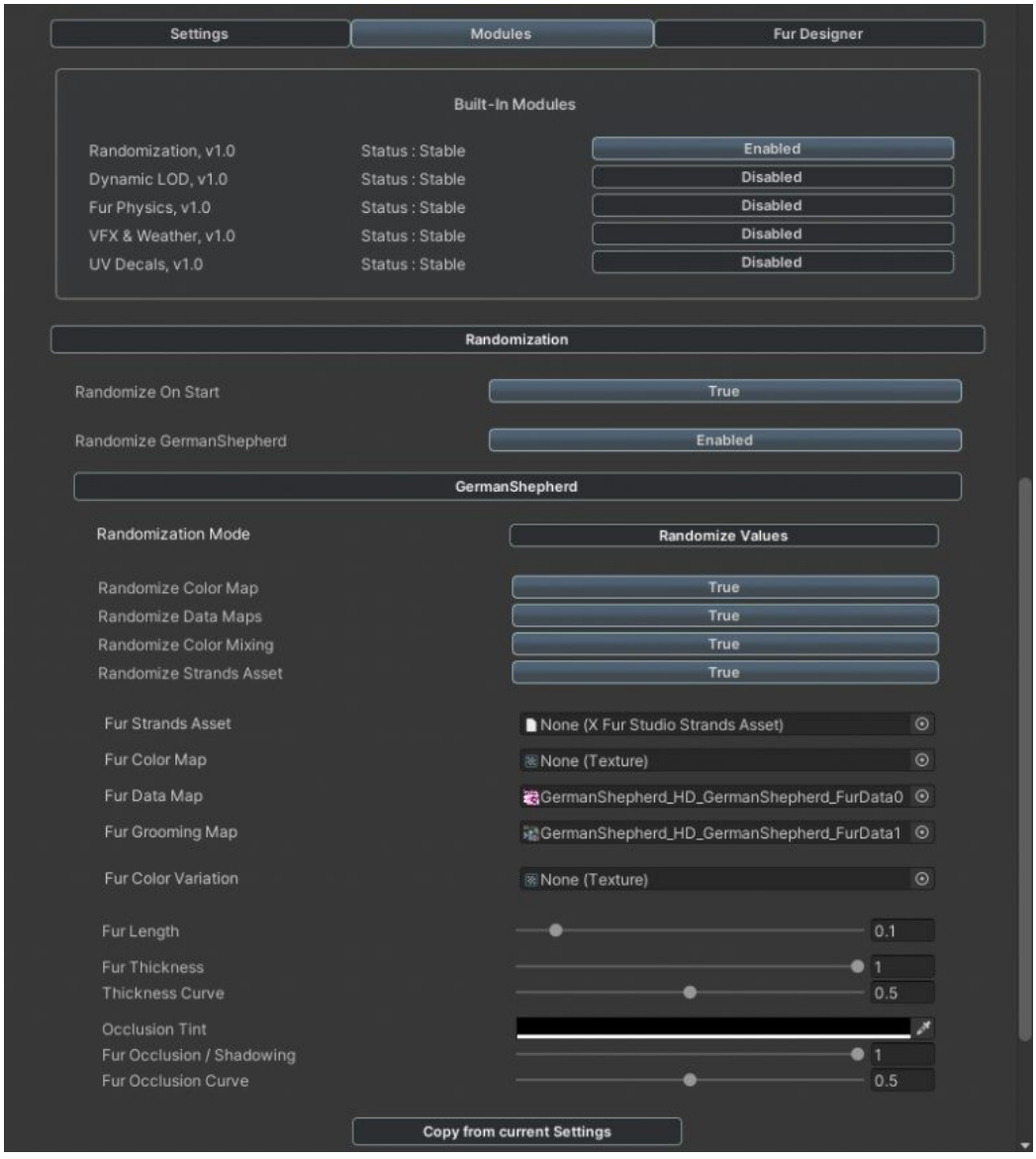
The Decals module allows you to project additional textures over your fur. Up to four decals per material can be applied and each one of them can be layered and mixed in overlay, additive or multiplied mode. These decals are projected using the main UVs of the character. In practical terms, it is similar to adding layers to a photoshop image in a very performant way.

Decals can be swapped at runtime since they are added in a non-destructive way. They can have different offset and tiling values for each decal, allowing great flexibility when customizing your characters.

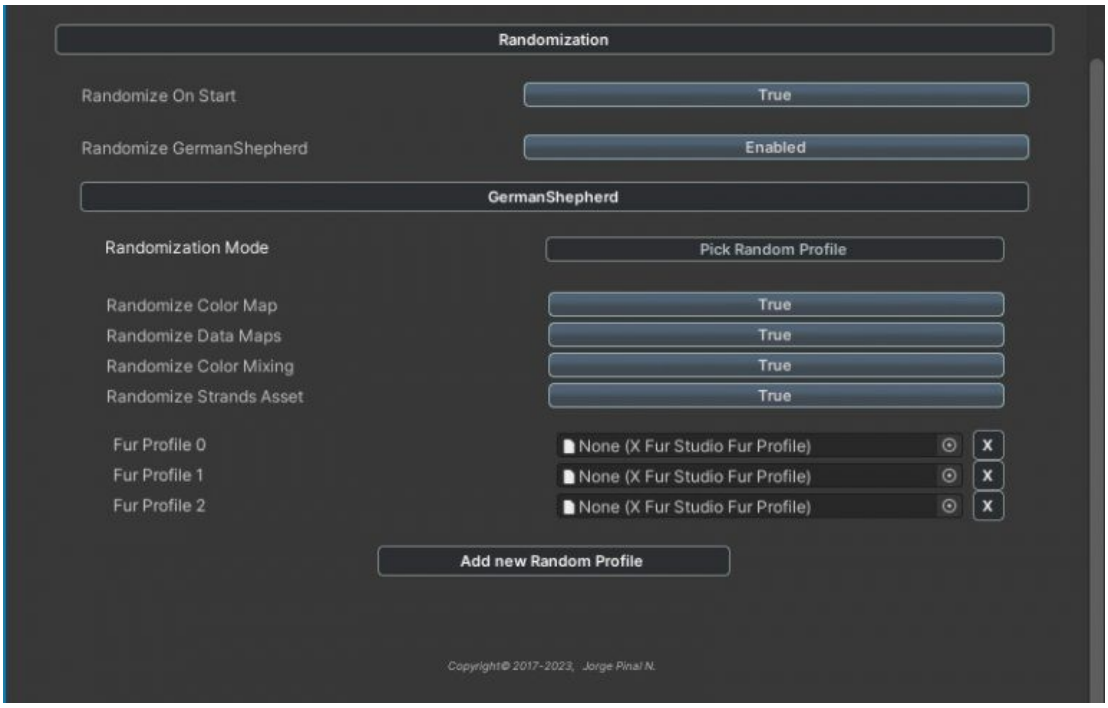
Randomization Module

The randomization module allows you to generate endless variations for your characters by manipulating all fur parameters at runtime within certain specific rules. The randomization module can be run in three different modes: **Randomized values, Pick Random Profile, and Randomize Values and Profiles**. The randomization can be triggered automatically on the Start function or manually through code.

When using the Randomize Values mode you can provide a series of alternative values for most fur parameters on a per-material basis. When active, the Randomization module will generate a new variant by selecting random values for each parameter that are located between their original values and the alternative values provided by you.



When using the Pick Random Profile mode, a list of Fur profiles is provided to the module for it to pick one at random. If you select the Randomize Values and Profiles mode each parameter will be randomized between the values of the original parameters and those of a random profile from the list.

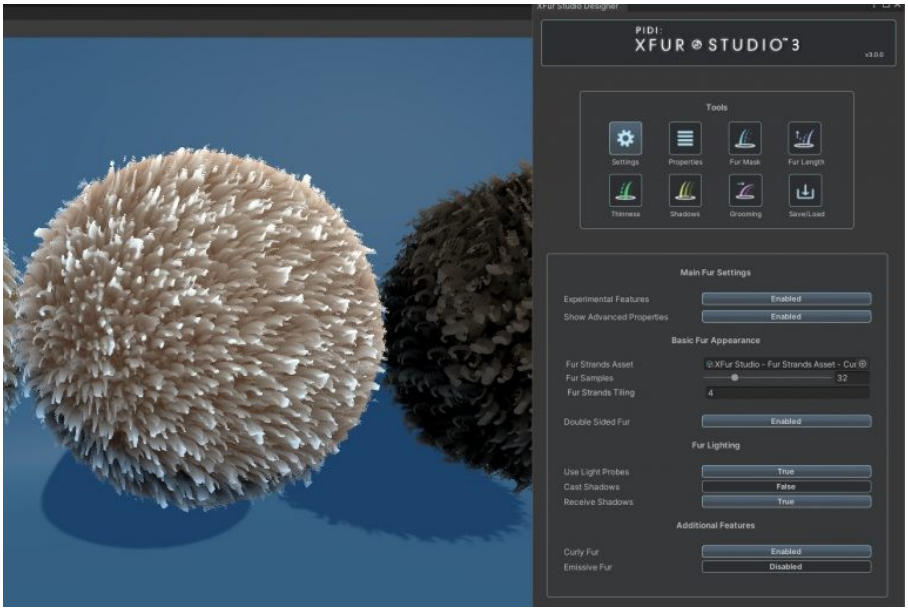


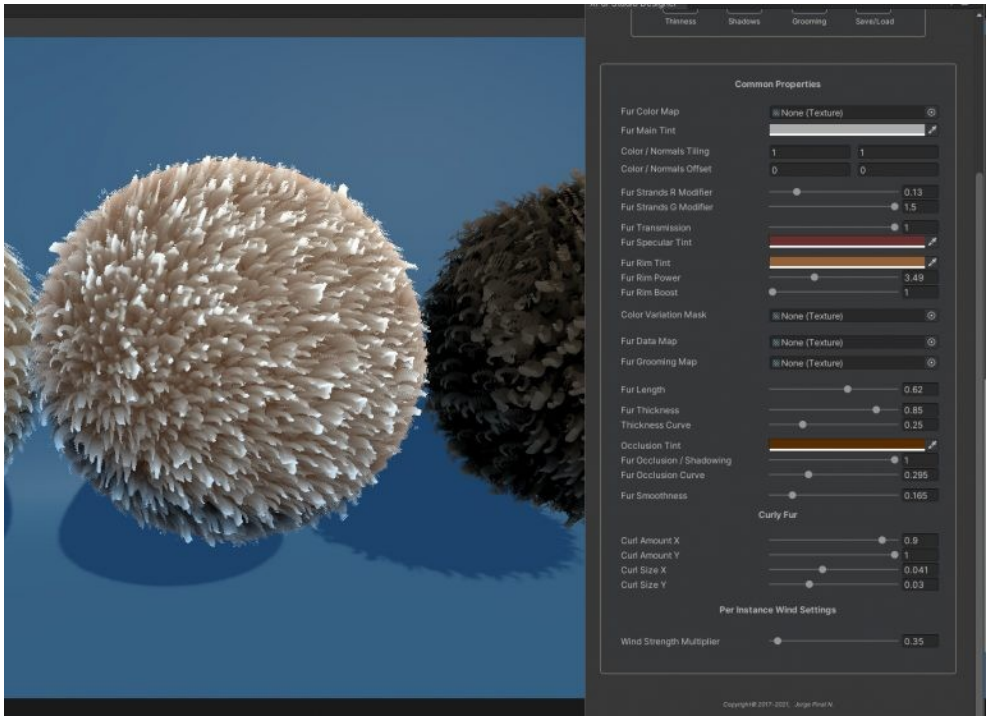
For any randomization mode, you can select whether to randomize only the general fur parameters or also the different textures and even the Strands Asset, which can allow you to have an almost infinite number of variations.

Curly Fur

The current curly fur implementation allows for manual adjustments of curl amount and size on both the X and Y axes and works with procedural and manual fur strand maps, the latter producing often times better results.

To enable the Curly fur parameters simply enable Curly fur in the Settings tab of XFur Studio Designer, and then head to Fur Properties to adjust it.



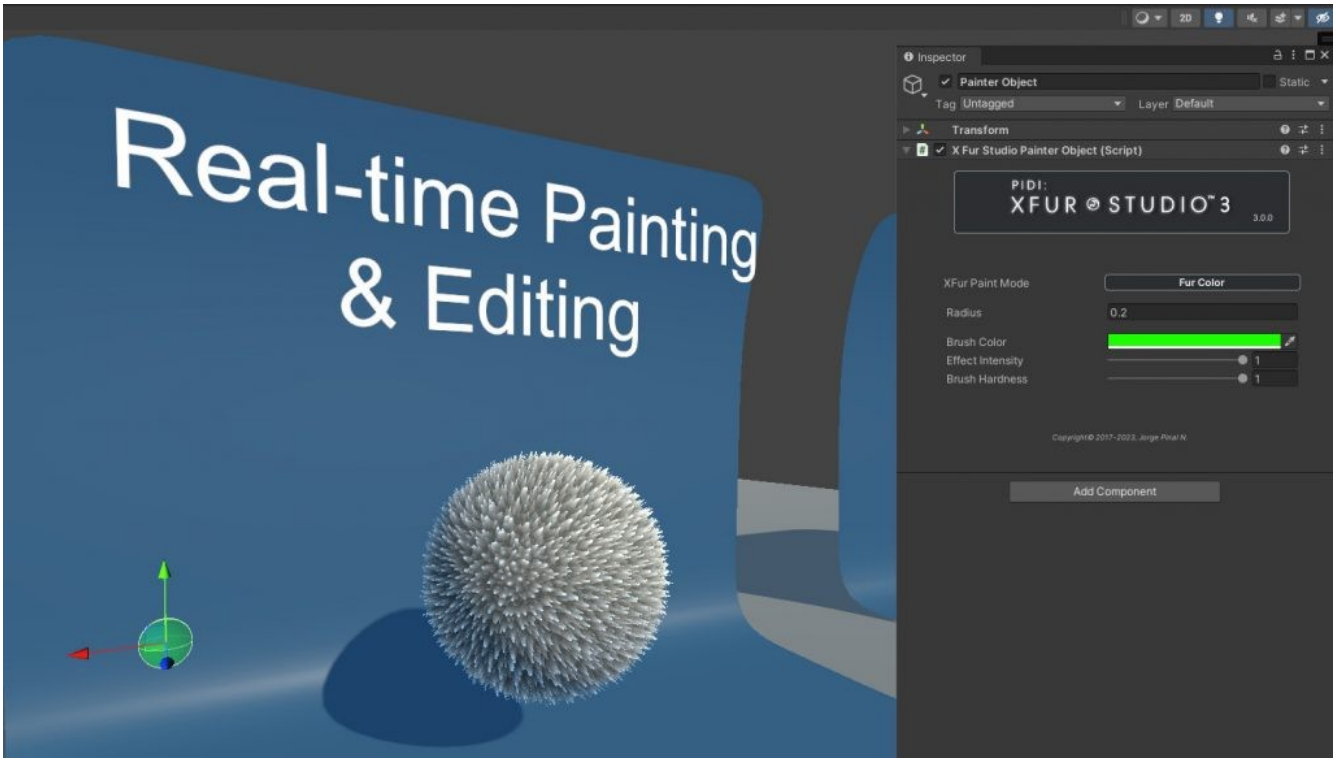


The available parameters control the number of waves / curls to be generated on the X and Y axes as well as their size. By playing with these parameters, you can create many different kinds of curly fur, from thin and well defined “spring-like” curls to wool-like materials, etc.

Curly fur is compatible with all XFur modules including physics and weather FX.

Painter Object

Painter Objects are extremely useful “Virtual Brushes” that allow you to modify the appearance of the fur at runtime in a similar fashion to XFur Designer. This is a feature that was very often requested in XFur Studio 1.x and XFur Mobile and lets you shave, paint or trim the fur using the Painter Object as a three dimensional, spherical brush.



Depending on the Paint Mode selected the UI of the Painter Object will automatically adjust itself to expose the right parameters to use, either a Brush color, effects intensity, whether to invert the painting mode (when painting the Fur Mask for example, it allows you to either add or shave away fur) etc.

To add an XFur Painter Object to your scene, simply create an empty game object and add the **XFurStudioPainterObject** component to it.

With future updates, additional Painter Object shapes and features will be added to allow for a more flexible integration with your projects and to expand what is achievable in XFur Studio 3 without any coding experience.

API Reference

Basics

To access any of the different XFur Studio 3 classes the first step is to properly include the corresponding namespace XFurStudio3. This ensures that there are no conflicts between XFur Studio 3 classes and any other scripts in your project. Since XFur Studio 3 uses custom assembly definitions to contain its classes, you must make sure to reference the correct assemblies from your own.

```
using UnityEngine;
using XFurStudio3;

public class MyClass:MonoBehaviour{
    //Your code goes here
}
```

While all the public accessible code meant to be available to your game is extensively commented and has contextual “<summary>” declarations to ensure in-line help pops out in your IDE, this section of the manual will cover the basics of interacting with XFur Studio 3 through code as well as using the brand new XFurStudioAPI

Your main point of access for the whole XFur Studio 3 code should always be the XFurStudioInstance component. To get a reference to the component use a standard GetComponent<T>() call from within your script like you would for any other component in Unity.

```
using UnityEngine;
using XFurStudio3;


public class MyClass:MonoBehaviour{

    public XFurStudioInstance xfur;

    void Start(){
        xfur = GetComponent<XFurStudioInstance>();
    }
    //Your code goes here
}
```

Once you have this reference, you are ready to access all the settings and features from XFur like we will see in the following sections.

XFur Studio Instance

There are several useful functions within an XFur Studio Instance that will allow you to further control the way the fur works as well as what you can do with your characters. As we have explained before, Fur Profile Assets are a great way to share and copy the settings and parameters from the fur of a character's material. Internally, these properties are stored in a class of type XFurTemplate.

The XFurStudioInstance component offers you functions to get and set these XFurTemplates on a per-material basis:

GetFurData

Retrieves the settings and properties of a fur material in use by this instance as an XFur Template. All properties are copied.

```
//"index" - The index of the material from which we will retrieve the properties
//"furTemplate" - The template from where the properties will be copied

public void GetFurData( int index, out XFurTemplate furTemplate )
```

SetFurData

Assigns an XFur Template with fur properties to the given material in this instance.

```
//"index" - The index of the material to set
//"furTemplate" - The template from where the properties will be copied
//"replaceColorMap" - Whether the fur color map should be set or left unchanged
//"replaceDataMaps" - Whether the fur data maps should be set or left unchanged
//"replaceColorMix" - Whether the fur color variation map should be set or left unchanged
//"replaceStrandsMap" - Whether the fur strands asset should be set or left unchanged

public void SetFurData( int index, XFurTemplate furTemplate, bool replaceColorMap = false,
bool replaceDataMaps = false, bool replaceColorMix = false, bool replaceStrandsMap = false )
```

SetFurDataFull

Assigns an XFur Template with fur properties to the given material in this instance. All properties and settings are overwritten.

```
//"index" - The index of the material to set
//"furTemplate" - The template from where the properties will be copied

public void SetFurDataFull( int index, XFurTemplate furTemplate )
```

Whenever fur properties are modified through code these changes are not applied nor visible immediately. If the XFurStudioInstance has its autoUpdateMaterials variable set to true (the default behavior) the changes will be visible in an instant. However, if this variable is set to false or if you want to see the changes immediately you must push these changes and perform a manual update.

UpdateFurMaterials

Applies all changes to the fur properties across all materials and forcefully updates any modules that work on the render loop.

```
//"forceUpdate" - Whether to force the update regardless of the autoUpdateMaterials settings

public void  UpdateFurMaterials( bool forceUpdate = false )
```

Below you can also find a list of all the public variables available in XFurStudioInstance that you can access and modify at runtime.

Variable Type	Variable Name	Variable's Purpose
XFurRenderingMode	RenderingMode	The rendering method to be used on this XFur Instance
bool	FullForwardMode	Enables / disables full forward lighting compatibility (point lights) mode
bool	AutoUpdateMaterials	Whether the fur properties will be updated and applied automatically
float	UpdateFrequency	The interval of time (seconds) between each automatic materials update
XFurTemplate[]	FurDataProfiles	The Fur Template profiles assigned to each material (Read Only)
XF3_Random	RandomizationModule	The randomization module assigned to this instance (Read Only)
XF3_LOD	LODModule	The dynamic lod module assigned to this instance (Read Only)
XF3_Physics	PhysicsModule	The physics module assigned to this instance (Read Only)
XF3_VFX	VFXModule	The vfx / weather module assigned to this instance (Read Only)
XF3_Decals	DecalsModule	The decals module assigned to this instance (Read Only)
XF3_WindZone	WindZone	A global wind zone affecting all XFur Studio Instances (static)

Fur Properties

All fur properties to be edited through code must be accessed through the XFur Studio Instance component by accessing the internal XFur Template (the struct containing all the fur properties for said materials) by its material index. The array of XFur Templates inside the XFur Studio Instance is called FurDataProfiles and it should be accessed as follows :

```
using XFurStudio3;
using UnityEngine;

public class MyClass:MonoBehaviour{

    void Start(){
        XFurStudioInstance xfurInstance = GetComponent<XFurStudioInstance>();
        int matIndex = 0;
        //Will print to the console the current length of the fur in the material with index 0 for
the
        //XFurStudioInstance component attached to this GameObject
        Debug.Log( xfurInstance.FurDataProfiles[matIndex].FurLength );
    }

}
```

Below you can find a list of all the accessible properties within each Fur Data Profile element.

Variable Type	Variable Name	Variable's Purpose
bool	CastShadows	Whether this fur material will cast accurate shadows
bool	ReceiveShadows	Controls the maximum length of the fur
int	FurSamples	The amount of samples / passes used on this fur material
float	FurUVTiling	The tiling applied to the Fur Strands map
float	FurLength	Controls the maximum length of the fur
float	FurThickness	Controls the thickness of the fur as it goes towards the tips
float	FurThicknessCurve	Controls the variation of the thickness from root to tips
float	FurOcclusion	Controls the overall self shadowing / occlusion in the fur
float	FurOcclusionCurve	Controls the strength of the shadowing over the strand
float	FurRimPower	Controls the thickness of the rim lighting outline
float	SelfWindStrength	The local multiplier for the overall wind simulation strength
float	SkinSmoothness	The smoothness of the "skin" pass on Basic Shell

Variable Type	Variable Name	Variable's Purpose
		shaders
Color	FurMainTint	The final tint to be applied to the fur
Color	FurColorA	The color to be blended by the red channel of the Color Variation Map
Color	FurColorB	The color to be blended by the green channel of the Color Variation Map
Color	FurColorC	The color to be blended by the blue channel of the Color Variation Map
Color	FurColorD	The color to be blended by the alpha channel of the Color Variation Map
Color	FurShadowsTint	The tint of the occlusion / self-shadowing effect
Color	FurRim	The color used for the Rim lighting effect
Color	FurEmissionColor	The color used for the emission channel when Emissive Fur is enabled
Color	SkinColor	The final tint applied to the "skin" pass of Basic Shells shaders
Texture	FurColorMap	The texture containing the color to be applied to the fur
Texture	FurData0	The texture that controls fur coverage, length, occlusion and thickness
Texture	FurData1	The texture that controls fur grooming direction (RGB) and stiffness (A)
Texture	FurEmissionMap	The texture that controls the emission channel of the fur
Texture	SkinColorMap	The color map of the "skin" pass on the Basic Shell shaders
Texture	SkinNormalMap	The normal map of the "skin" pass on the Basic Shell shaders
XFurStudioStrandsAsset	FurStrandsAsset	The fur generation map (procedural or Texture) stored in an asset

Painting API

The XFurStudioAPI is a static class that allows you to dynamically paint and edit the fur appearance at runtime with virtual, world-space spherical brushes through code. The Painter Objects included with XFur Studio 3 are based on the XFurStudioAPI as is XFur Studio Designer, meaning that you have the same powerful set of tools used to customize the fur within the editor available at runtime to expand massively the number of things you can do with XFur within your game.

Using the XFurStudioAPI is very simple and in this last section of the manual we will cover its main functions.

Groom

The Groom function, as its name implies, allows you to pass a world-space direction to an XFur Instance's specific fur material and apply deformation in the form of grooming to the fur in that same direction. This world space direction is internally transformed to mesh based coordinates to ensure that the direction stays the same even in animated characters

```
//"target" - The target XFur Studio instance whose fur will be groomed
//"matIndex" - The index of the material to be edited
//"brushCenter" - The center of the brush sphere, in world coordinates
//"brushNormal" - The normal / direction in which the brush is looking, in world coordinates
//"brushSize" - The size of the brush sphere in world units
//"brushOpacity" - The opacity of the brush as a float between 0 and 1
//"brushHardness" - The hardness of the brush as a float between 0 and 1
//"groomDirection" - The direction in world coordinates in which the fur will be groomed
//"invert" - If true, grooming data will be removed rather than applied

public static void Groom( XFurStudioInstance target, int matIndex, Vector3 brushCenter, Vector3 br
ushNormal,
float brushSize, float brushOpacity, float brushHardness, Vector3 groomDirection, bool invert = fa
lse )

//Example :

public Transform myBrushObject;

void OnApplyGrooming(){
    //Will groom the fur upwards wherever the myBrushObject transform intersects the xfurInstance
object within
    //a 0.25f units radius
    XFurStudioAPI.Groom( xfurInstance, 0, myBrushObject.position, myBrushObject.forward, 0.25f, 0.
35f, 0.5f,
    Vector3.up );
}
```

Paint

The Paint function allows you to modify all the other aspects of the fur that are driven by textures, including its length / thickness / occlusion (via the FurData0 map), its color (via the FurColorMap), its color variation (via the ColorVariationMap) and even the output of the VFX module to add effects such as blood, snow or rain / wetness at runtime.

```
//"target" - The target XFur Studio instance whose fur will be groomed
//"painterMode" - The data that will be modified
//"matIndex" - The index of the material to be edited
//"brushCenter" - The center of the brush sphere, in world coordinates
//"brushNormal" - The normal / direction in which the brush is looking, in world coordinates
//"brushSize" - The size of the brush sphere in world units
//"brushOpacity" - The opacity of the brush as a float between 0 and 1
//"brushHardness" - The hardness of the brush as a float between 0 and 1
//"brushColor" - The color of the brush. For fur data values and masks use either white or black
//"brushTexture" - Optional brush texture ( functionality not fully implemented yet)

public static void Paint( XFurStudioInstance target, PaintDataMode painterMode, int matIndex, Vect
or3 brushCenter,
Vector3 brushNormal, float brushSize, float brushOpacity, float brushHardness, Color brushColor, T
exture brushTexture )

//Example :

public Transform myBrushObject;

void OnApplyRedPaint(){
    //Will paitn the fur red wherever the myBrushObject transform intersects the xfurInstance obje
ct within
    //a 0.25f units radius. Since brushTexture is not fully implemented yet it is extremely import
ant to
    //pass this value as Texture2D.whiteTexture
    XFurStudioAPI.Paint( xfurInstance, XFurStudioAPI.PaintDataMode.FurColor, 0, myBrushObject.posi
tion,
    myBrushObject.forward, 0.25f, 0.35f, 0.5f, Color.red, Texture2D.whiteTexture );
}
```

With these two functions you can easily create all kinds of interactions between your game's world, its players and XFur Studio™ 3 in ways that were not possible before. An almost limitless amount of gameplay possibilities are available thanks to these API commands and the new modules system, Painter Objects, WindZone objects and more. With XFur Studio™ 3 you have the most complete, efficient and advanced fur simulation system for Unity in any rendering pipeline you use, and for any device you develop.

If you have any questions, suggestions or technical issues related to this tool please don't hesitate in contacting us to our support email. For support requests, please make sure to include your invoice / proof of purchase with your invoice number visible, the exact Unity version & rendering pipeline you used as well as the exact version of XFur Studio you are using for your project as well as all the relevant information and screenshots that may help us reproduce the issue.

We thank you for choosing XFur Studio 3 for your project and hope that it will assist you in creating all sorts of amazing creatures and worlds.

Troubleshooting

There are some common issues that you may experience when working with XFur Studio™ 3. In this guide (which will be updated as time goes by) we will teach you how to solve some of them in an efficient manner.

Fur strands look strange

The most likely cause is an error with the model's secondary UV map. If your model has a secondary UV map that is not properly generated then the strands will be projected in odd ways. In a similar fashion, if there is a secondary UV map but the model has not been unwrapped onto that second map then the character will not show any fur at all. In most cases, the easiest way to solve this is to add a correct secondary UV layout for your model in your preferred 3D app or, if you have no access to one, to enable the "Generate Light Map UVs" option in Unity when importing the model.

Fur looks too short

In most cases this error is caused by your model having the wrong scale. Select your model's renderer and check its transform. If the scale is not 1,1,1 and it is instead a value larger than 1 then it will produce fur that looks too short. A scale lower than 1 may produce fur that looks overly long.

Painting does not work / does not detect model surface

This is usually caused by errors in the rotation / scale of the model. If your model's renderer has the wrong scale / rotation (most Blender exported models have a 90 rotation on the X axis that has to be corrected by hand within Unity if it wasn't accounted for when exporting) then the invisible collider generated so that you can paint it using XFur Designer will not be in the correct rotation / scale and the position you are trying to paint on your model will not have a matching one in the collider. Making sure that your model's renderer has a scale of 1,1,1 and a rotation of 0,0,0 is required in order to paint them without issues.

Grooming symmetry is not working as expected

In most cases this is due to the model or its root object being rotated in the scene. For better results, ensure that your model is rotated at 0,0,0 when grooming it, especially if using the symmetry mode.