
PIDI
GAME DEVELOPMENT FRAMEWORK ™
BY IRREVERENT SOFTWARE™

PIDI

GAME DEVELOPMENT FRAMEWORK™

BY IRREVERENT SOFTWARE™

XFur Studio

USER MANUAL

Index

PIDI – XFur Studio
Introduction

PIDI- XFur Studio is a complete system of tools and shaders designed to add fur easily to any model in Unity. The shaders included with our package are packed with many features not seen in most other fur solutions, such as full PBR integration, physics, weather effects, randomization, full lights and shadows support or full deferred and post process effects support.

This package also includes XFur Painter, a tool that allows you to add fur and precisely control all its settings by painting directly over any 3D model inside the Unity Editor to then export the resulting fur maps in a format compatible with the shaders. No need for any external tool nor any complex edition process.

This shaders collection has been developed to provide the highest quality possible with a good performance. The high fidelity of the rendering techniques used as well as their complexity make it a AAA-ready solution for high end games.

This shader has not been designed nor is intended to be used with low end PCs nor mobile devices. While many optimizations have been made and a lot of work has been done to reduce its memory footprint and draw call requirements it is still best suited for newer and dedicated graphics cards. Using a dedicated graphics card you can render dozens of animated models using this shader without a significant performance loss.

This manual will explain the basics of the shaders and how to set it up to be compatible with either third party models (such as those bought in the Asset Store) or with your custom-made 3D assets.

It will also teach you the best practices on lighting and rendering of fur-enabled meshes to get the maximum performance from every scene, as well as how to use the XFur Painter tool with ease to quickly get fur maps ready to use.

Thank you for purchasing this tool, I really hope that it will help you to keep making amazing games!

Jorge Pinal, Irreverent Software™

1. XFur Studio : A new approach to fur rendering

XFur Studio is a complete solution for fur rendering in Unity 3D, aiming to provide all the functionality that your game may require from any fur-covered characters, from a high performance-high quality rendering and deep customization options to realistic physics behaviors, weather effects and randomization.

After XFur 1.0 and 1.1 the main goal has been to streamline the workflow of this toolset to make it as easy to use as possible while making it simpler to upgrade, polish and evolve.

The new XFur Studio is the result of all this work, including every single feature from previous releases while adding new exciting components through an easy to use and understand UI.

Before you start using the new XFur Studio it is necessary that you familiarize yourself with all the new methods, components, shaders and tools so you can take the maximum advantage of the package.

After this brief introduction of all the new features and changes we will cover all the general practices to have your characters covered in fur in no time, as well as some tips for best performance and a few advanced topics on how to extend the system and customize it.

I. Standard & Dynamic Shaders

There are 2 types of shaders included with XFur Studio 1.5 : Standard Shaders (with 4, 8, 16 and 24 samples support) and Dynamic Shaders (Dynamic Fur Pass and Dynamic Base Pass)

- **The Dynamic Shaders** are designed to support the newly added **Advanced Shadows.** They are very heavy to compute and produce a **considerable** drop in performance if used on highly detailed models or with many characters. Use them with precaution. Their shadows work without issues in both forward and deferred rendering.

Fig. 1 – Anisotropic-based shader from the new Dynamic branch with highly accurate shadows

- **The Standard Shaders** have the best balance between high quality rendering, good performance and an all around flexible workflow. Their only downside is that the shadows produced by them do not display the fur strands and, in forward mode, the fur does not receive shadows accurately.

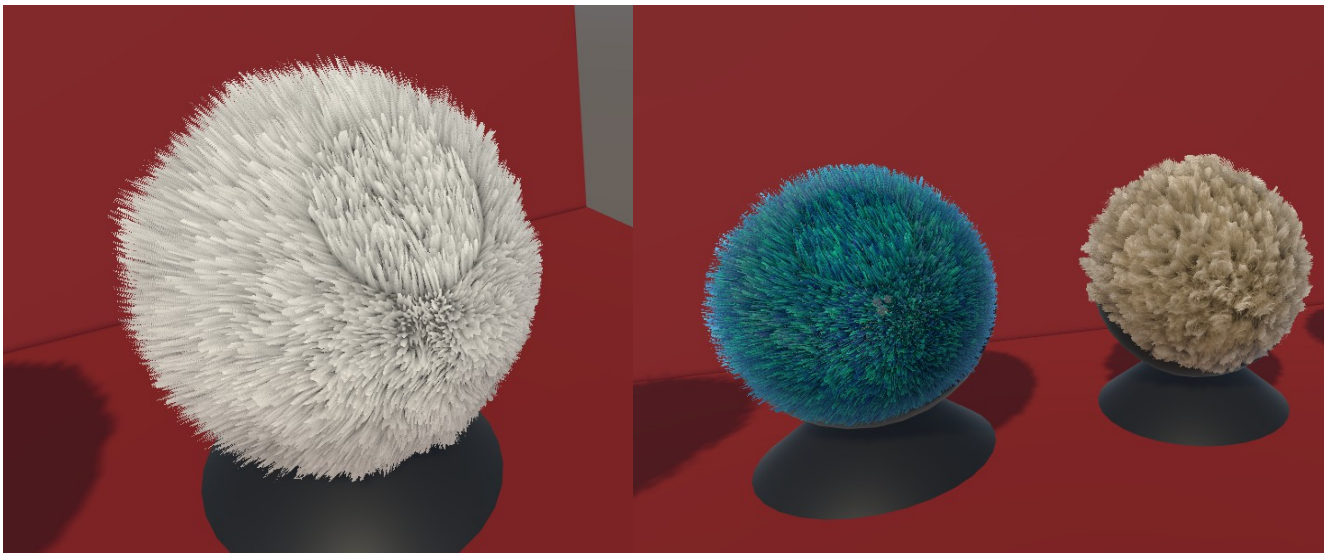


Fig. 2 – Improved fur strands on the new Standard Shader, improved fur generation, complex coloration patterns and the brand new "Curly" fur material (right)

While the old legacy shaders had different shader files for Triplanar, Anisotropic, Painter and Specular variants, the new **Dynamic** and **Standard** shaders contain all the variants within 6 shader files. These variants are easy to access and setup through toggles in their respective Uis, either directly on the material or through the XFur System component.

One of the biggest changes is how the fur strands themselves are generated and handled, as well as the way their color is applied. The old fur textures packed a basic noise map as a grayscale and this was used to generate the fur. The new fur textures pack 4 grayscale channels on a single texture controlling overall shape of the fur (volume), alternative patterns, color variation and fur noise.

Additionally, the new shaders have full support for double sided rendering at a very little performance cost, adding more realism to the final rendering.

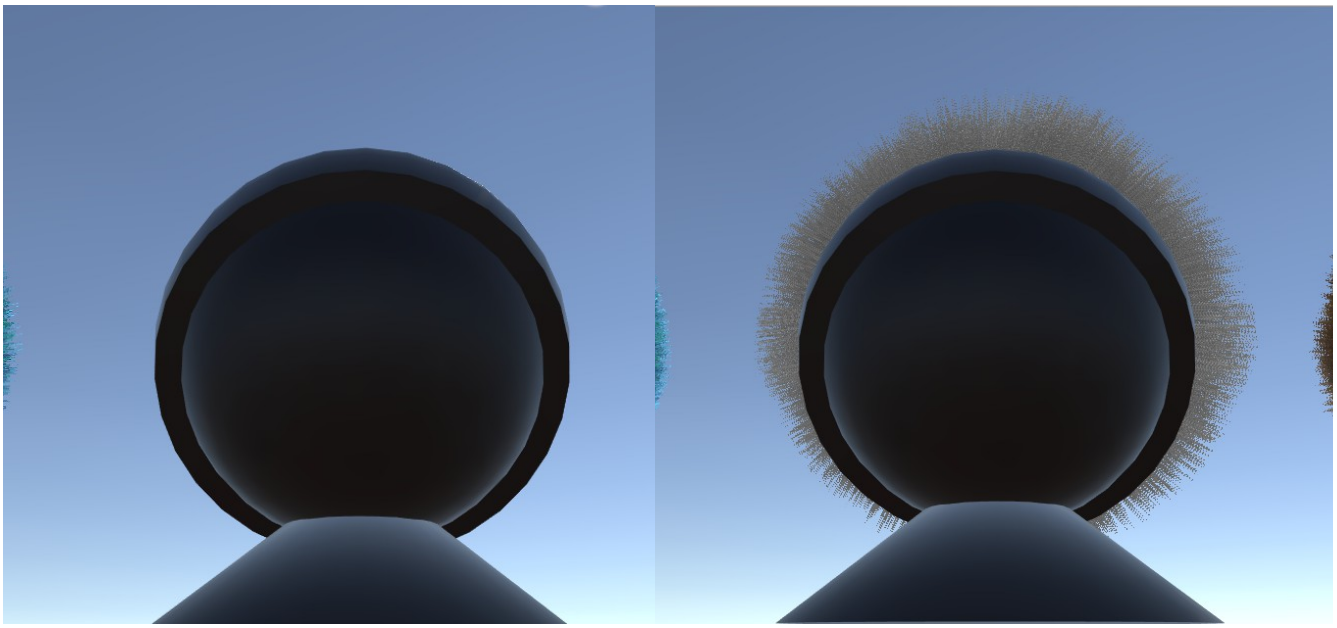


Fig. 4 – Backside of the material sphere without (left) and with (right) double sided rendering enabled.

Due to all the new features included as well as the many new customization options, all shaders are slightly more expensive to render when using no features than the old ones. However, the performance stays relatively the same even when using all the features thanks to the extensive use of variants that disable/enable parts of the shader depending on the use.

Triplanar projections have become more expensive to calculate due to the additional data packed on each texture. While this performance cost is not noticeable with a few on-screen characters, it is still recommended to generate a custom secondary UV map whenever possible to reduce the performance overhead on the graphics card.

The standard shaders can be added to any material and assigned to any model in the same way as in previous versions. However, to take advantage of all the new features it is necessary to add the XFur System component to the renderer as well. We will cover this in more detail in the next chapter.

II. XFur System Component and Databases

There are two main, core components on XFur Studio that control the behavior and functionality of everything else : the **XFur System Component** and the **XFur Database.**

The **XFur System** component controls the fur's appearance and behavior as well as all the different modules and effects that will be used. LOD, Physics, special effects, randomization, shader features, maps, colors, properties etc. can be found on this component.

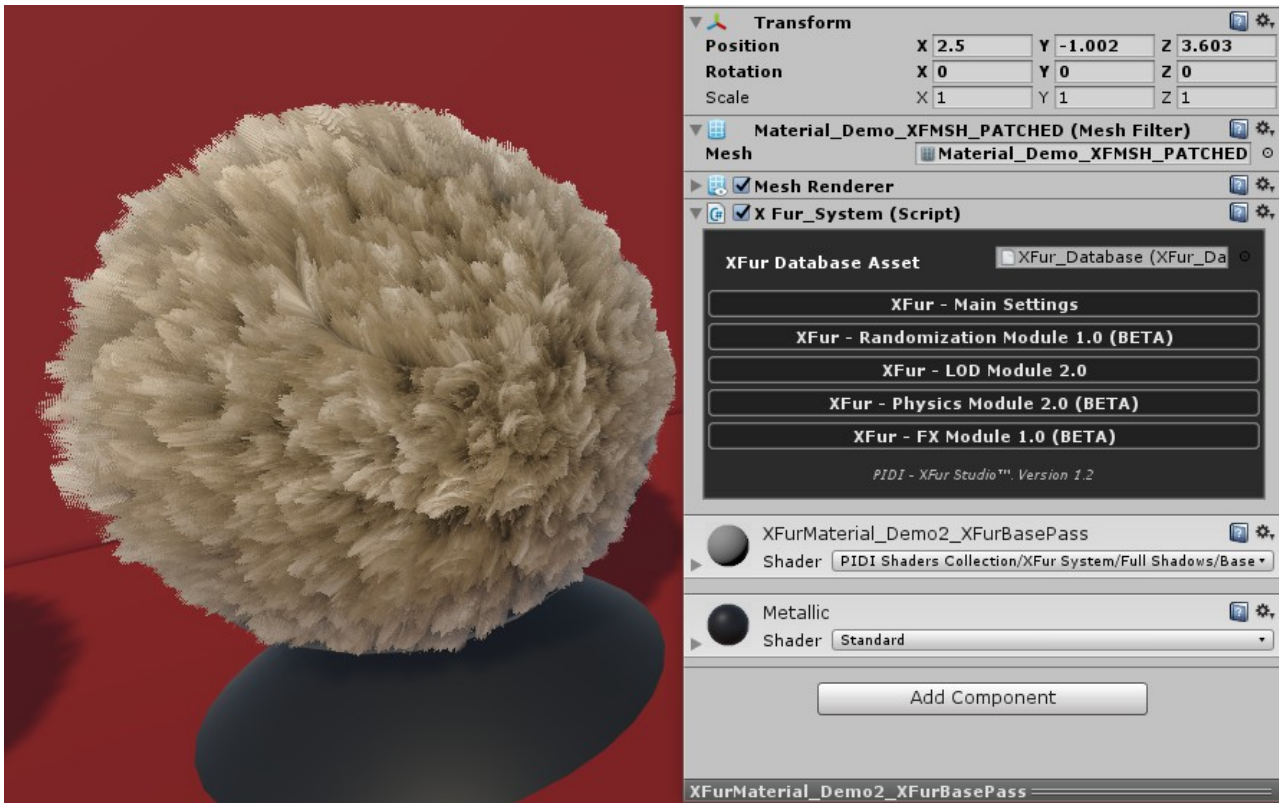


Fig. 5 – Basic view of the XFur System component

The XFur System design is instance-oriented. This means that all the properties, behaviors and modules work on an instance basis instead of a material basis. You can share the same material across countless instances and have different parameters and behaviors applied to each one of them without any issues.

Objects with multiple fur-based materials are also supported by the system but the per-instance parameters are turned off automatically in Unity versions prior to Unity 2018.1 due to limits on those engine versions, relying on the actual material parameters instead. The rest of the behaviors (physics, lod, FX, etc) are all handled on an instance basis in all Unity versions.

The XFur System component **requires** an XFur Database asset as a reference to be able to work. It will also allow you to automatically upgrade legacy materials so they require minimal tweaking before you can use them with the new system.

The **XFur Database** is a very special data type that stores the references to all the available shaders as well as all the patched and generated geometry used by XFur to ensure that your original meshes are not modified in any way, shape or form.

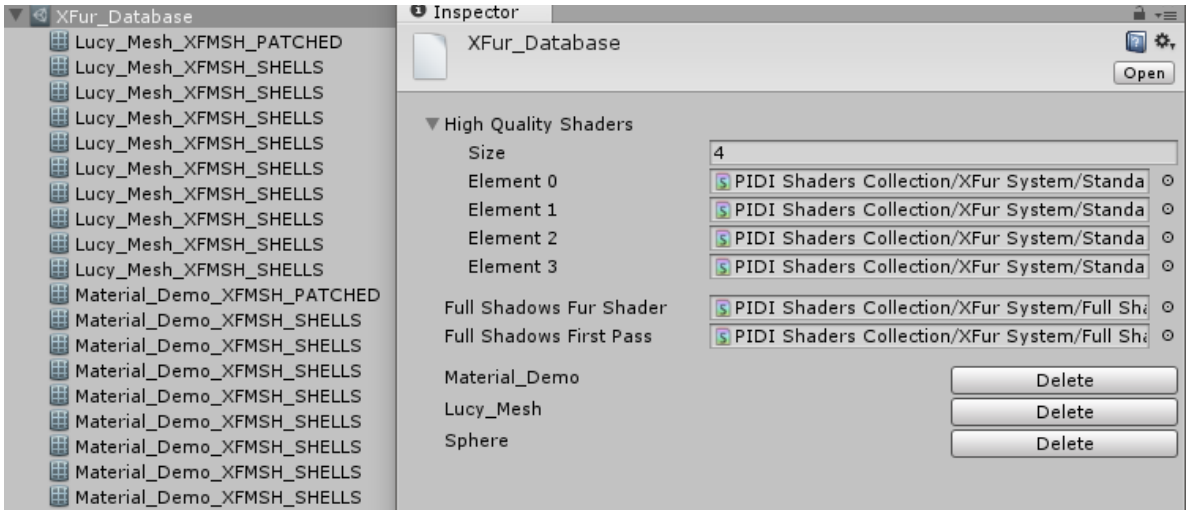


Fig. 6 – Basic view of the XFur Database provided with the package

XFur Studio uses a clever mix of procedural and baked information to perform all its functions. This way, a lot of the information required for physics and other usually heavy tasks is pre-calculated while developing the game leaving it 100% ready for runtime, avoiding long loading times and pauses for your end users. For this, XFur generates modified copies of your original meshes and patches them with all the additional information required by the shaders. These meshes are then stored on the main **XFur Database** so they can be referenced by similar characters that share the same original mesh without having to perform the patching and heavy algorithms more than once.

While an example database is provided with this package, you MUST create your own database asset before using XFur Studio and store it somewhere outside the XFur Studio folder to prevent it from being overwritten when updating this asset to future versions. Do not use the provided database asset for any final project. You can create your own databases by going to the menu : Assets/Create/XFur Studio/Modules/Database Asset.

Once created, assign the high quality shaders and the Full Shadows Fur Shader and Full Shadows First Pass Shader in a way that they match the values of the provided database. Once this is done, your database is ready to use.

As an alternative method, you can duplicate the provided database and simply click on the "Delete" button for each one of the assigned meshes, which will leave the database clean and ready to use.

Both databases and the XFur System component will be covered in detail in the next chapters of this manual.

III. XFur Painter 2.0

XFur Painter has received a considerable update between versions with completely redesigned Uis and developed around the new XFur System. With better support for multiple materials, interrupted and resumed work, more responsive and requiring almost no setup to work it represents a huge improvement over version 1.0.

There is no need to manually add a mesh collider, you can add a full rigged model to the scene without issues, there is no need to manually define materials and meshes to the Painter component and most other steps have also been fully automated.

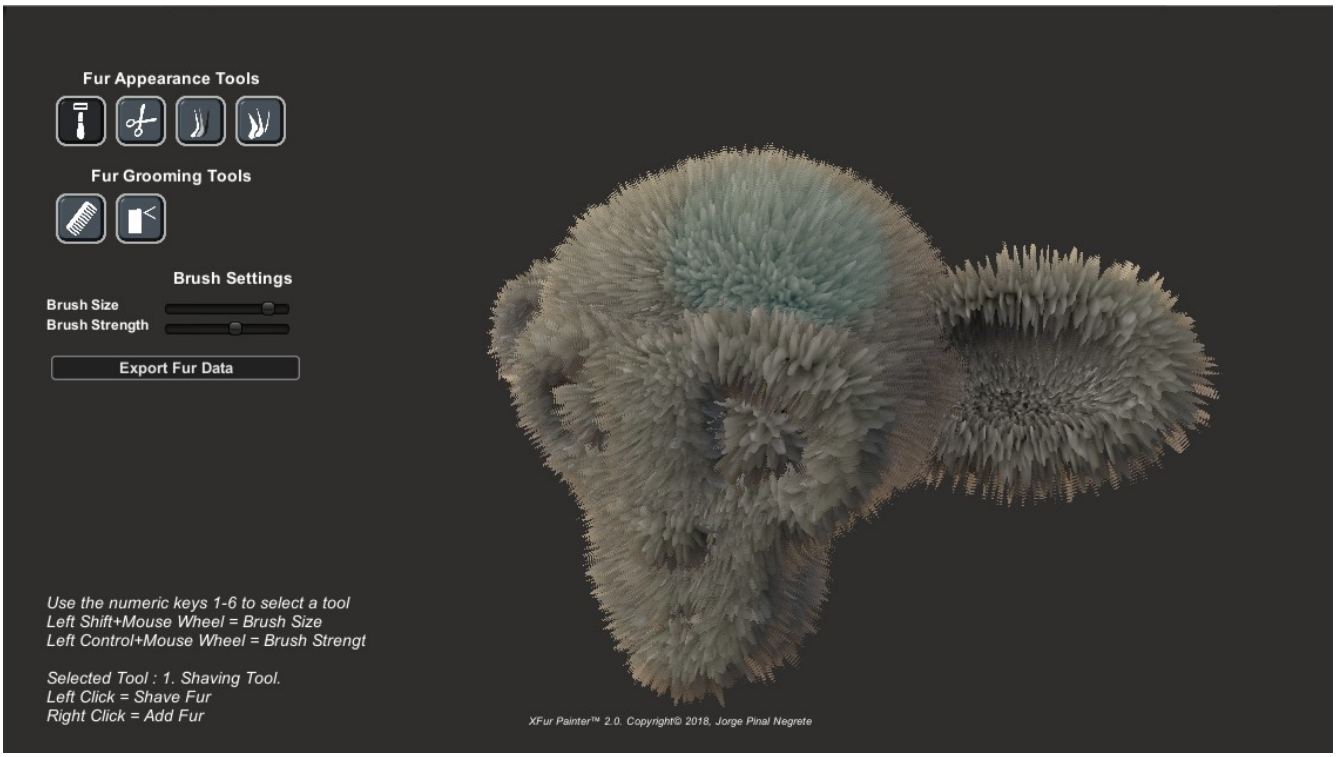


Fig. 7 – The new XFur Painter 2.0

While the controls have been kept consistent with the previous version to make the learning curve almost non-existant, there are a few differences that will be covered in depth in the dedicated chapter further ahead in this manual.

Now that we have covered all the main components of the new XFur Studio 1.2 we are ready to dive into the more detailed topics and learn how to effectively use each feature included with the system and how to prepare your models for usage with this system.

2. XFur Studio 1.7 : System Workflow

I. Preparing your Assets for XFur Studio

XFur Studio doesn't have any particular requirements for the assets to be used with any of the included shaders and it can automatically take care of patching the meshes as needed to support all fur features. Meshes without a secondary UV channel can be used without issues by enabling triplanar projections on the material, which will take care of projecting the fur in a uniform way.

However, due to the performance impact of triplanar projections on the graphics card, it is heavily recommended that you use a 3D modeling app to generate accurate UV maps for your model on the secondary UV channel. Usually, a cubic projection will be very effective for these purposes.

Make sure that your model has correct transformation values applied. This means that the mesh inside of Unity should display a 0,0,0 location, 0,0,0 rotation and 1,1,1 scale. Any other values in the transform of your mesh may cause issues with Xfur's functionality. The rotation and location can be set manually inside of Unity without issues, but the scale has to be fixed in your 3D modeling app.

II. Creating Fur Materials

The intended workflow for XFur Studio is to reduce the amount of materials as much as possible. While several versions of the Standard shader are provided at different sample amounts (4, 8, 16, 24 samples) you should not create more than 1 material variant. The XFur System component will take care automatically of managing the sample amount on your characters without you needing to generate additional materials manually.

To create a fur material, you first need to create a normal material as usual. Once your material has been created, select one of the new XFur shaders and the material will automatically display the custom shader editor used for all XFur Materials. **It is heavily recommended to always set your materials to use the Standard/High Quality/Twenty Four Samples shader as a starting point.**

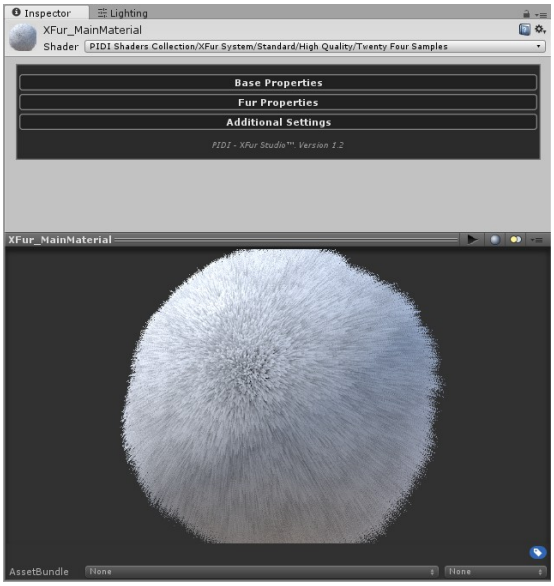


Fig. 7 – The new custom inspector for all XFur shaders

A newly created material will appear without any fur nor textures on it. To make the fur appear, open the **Fur Properties** tab and assign one of the provided XFur_Layer textures to the **Fur Gen Map** slot. Adjust the Fur Cutoff and Fur Thickness values until the fur has a correct appearance.

For better results, set the Fur Direction vector to 0,0,0,0 as in newly created materials it will not be used.

Now, let's explain in detail every single feature available on the material :

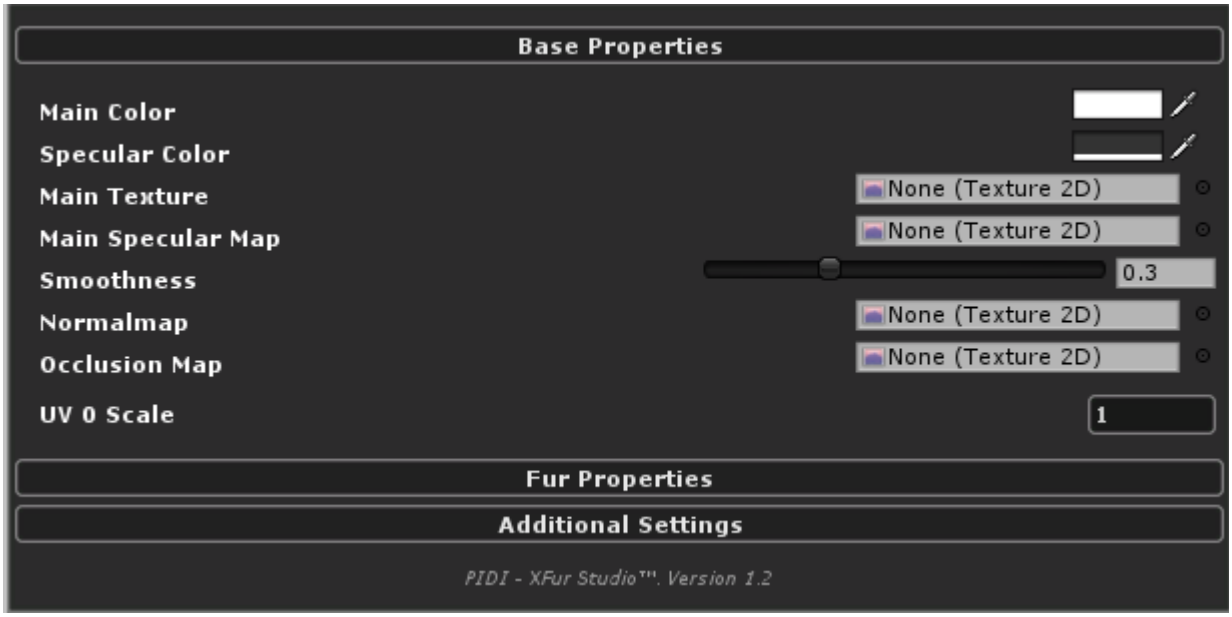


Fig. 8 – The new custom inspector for all XFur shaders. Base Properties tab

On the **Base Properties** tab you can control the properties that affect the skin layer of the material. The Main Color, Specular Color, Smoothness, Main Texture, Main Specular Map, Occlusion Map and Normal map work in the same ways as those in the Standard Specular Shader.

The UV 0 Scale controls the tiling of all of these textures, which all share the main UV coordinate set. This UV coordinate is also used for the main Fur Color Map to be assigned to this fur material.

Hovering your cursor over any of the inputs of this material will show a small tool-tip providing detailed information on what it is used for :

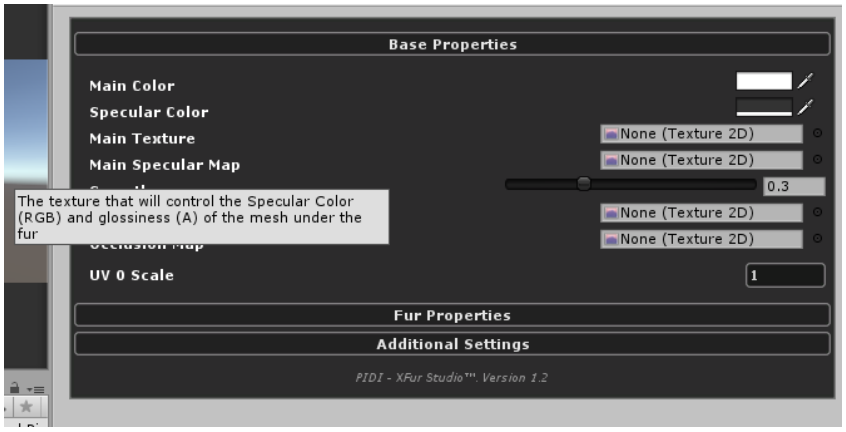


Fig. 9 – An example of one of the many tool tips and visual aids provided in every single UI of XFur Studio

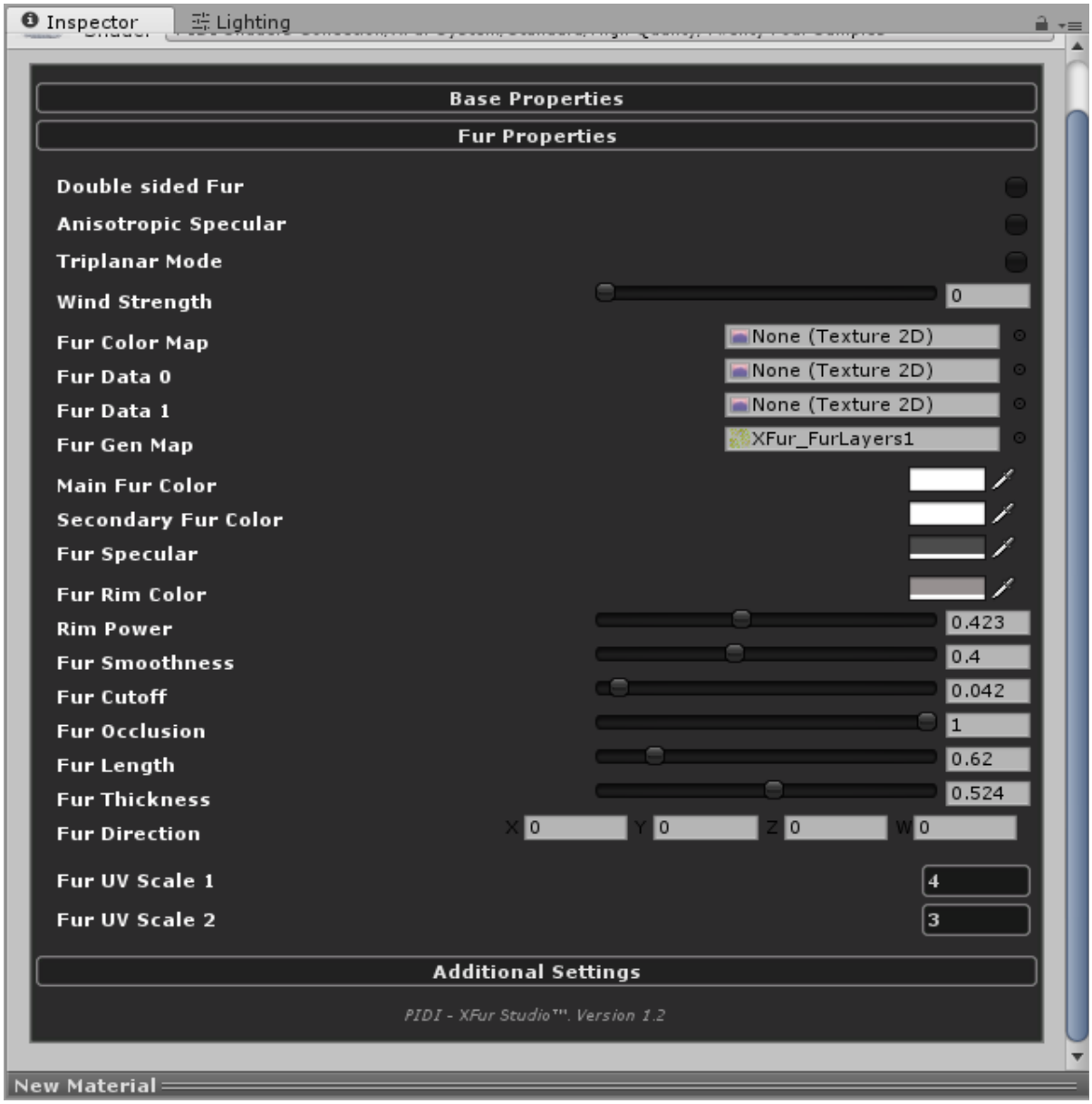


Fig. 10 – Fur properties tab

The Fur Properties tab contains most of the settings available on the material. With the first three settings you can set up the fur to be rendered in double sided mode, enable triplanar projections and add anisotropic highlights. When anisotropic highlights are enabled, an additional slider to control their offset is also available. When triplanar projections are enabled, and additional slider lets you control the triplanar scaling as well.

The **Wind Strength** slider controls the strength of the wind effect for this material. This way you can make the effect stronger or more subtle than the actual wind speed defined on the scene.

The **Fur Color Map, Fur Data0, Fur Data1** and **Fur Gen Map** are the maps required to correctly display and render your fur material. The Color Map is the actual color texture to be applied to the whole fur; Fur Data 0 is the texture that contains the Mask, Length, Occlusion and Thickness information; Fur Data 1 contains the grooming and stiffness information while Fur Gen Map is the actual Fur Noise map that

will generate the fur strands and their final appearance.

With XFur 1.2 coloring of fur has become more complex and rich, allowing you to mix two different colors with the variation between them depending on the blue channel of the XFur_Layers texture. The Main Fur Color and Secondary Fur Color will allow you to control this 2 color variation across the fur. The Main Fur Color is applied on most of the fur and is multiplied by the main Fur Color Map. The secondary fur color is mixed in an additive-pre multiplied way.

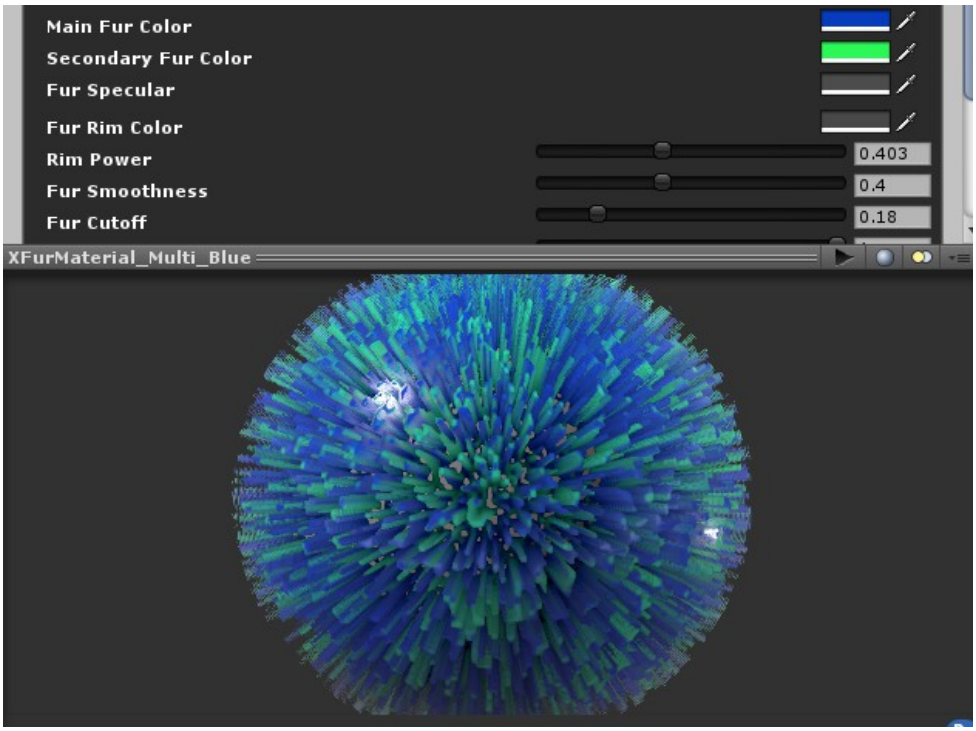


Fig. 11 – Advanced coloration in XFur 1.2

Fur Specular and **Fur Rim Color** control the specularity and the back-light effect over the fur. The Rim Power controls how much of the rim lighting will cover the mesh. Fur Smoothness controls how shiny will the fur look, and it is useful to simulate wetness. Fur Cutoff along with Fur Thickness give you precise control over how the fur strands are rendered and presented as well as on how thick or thin they will be.

Fur Occlusion controls the fur's self shadowing, to add some depth to its appearance. Fur Length and Direction are self explanatory. **Please be aware, however, than in order to groom your fur in Painter and have it work with Groom maps you need to set the Fur Direction vector to any value other than zero. For most cases, a value of 0.2,0.2,0.2,0.2 will work fine.**

The Fur UV Scale 1 and 2 control the density of the fur strands and the distribution of the color variation/overall shape of the fur distribution.

Fig. 12 – Additional settings tab

Finally, the **Additional Settings** tab contains a parameter for the FX noise texture, which controls the appearance and distribution of the special effects such as snow, water and blood over the materials.

The FX UV Scale parameter controls the tiling of this texture.

Since we will be using the XFur System component which will take care of all the fur settings and override the material properties, we can leave most settings here unchanged. Just set the Wind Strength and Fur Direction values to 0 and assign the XFur_FurLayers1 texture to the Fur Gen Map slot. Set the Fur Cutoff value to a low 0.1 and Fur Thickness to 0.3 and let's go to the next step.

III. Creating a Database

It is heavily recommended to use your own database to store all XFur related information instead of relying in the one provided with the package, as this one may be overwritten by future updates and delete your custom information in the process.

Creating a database is easy, however. Simply duplicate the database provided with the package (you can find it in the XFur folder, inside the XFur Studio Source/Database Module folder. It is the asset called XFur_Database. Once you have duplicated the asset by selecting it and pressing Ctrl+D, move it to a different folder on your project, change its name and click the Delete button on all three names that appear on it.

This database is a central component of the XFur System component and it is a MUST if you want to work with XFur Studio effectively.

Alternatively, you can create a database from the Assets/Create/XFur Studio/XFur Modules/Database Asset menu but in this way you would need to also assign the shaders manually to match the ones in the provided database.

IV. Adding the XFur System Component

Place your model on the scene and assign the XFur Material we just created to it. If you enabled triplanar projections on the material, the fur will look very wrong over your model's surface but do not worry, it will get fixed once we add the XFur System component. Make sure your model's mesh has a position of 0,0,0 and a rotation of 0,0,0 as well. The scale should be set to 1,1,1 to have accurate results.

Fig. 12 – A model before (left) and after adding a fur material (right)

Add the XFur System component in the same Game Object that has the Renderer Component (Mesh Renderer for static meshes and Skinned Mesh Renderer for animated ones). The component will display an error message asking you to assign a XFur Database asset to it. Let's assign the database we created to the only slot available on the UI.

Fig. 13 – Default error message when using XFur System without a database

Depending on the complexity of your model, you will need to wait a few seconds before anything happens. XFur is creating a copy of your model, patching it and storing it in the database. Once everything is done, the XFur System UI will be fully visible.

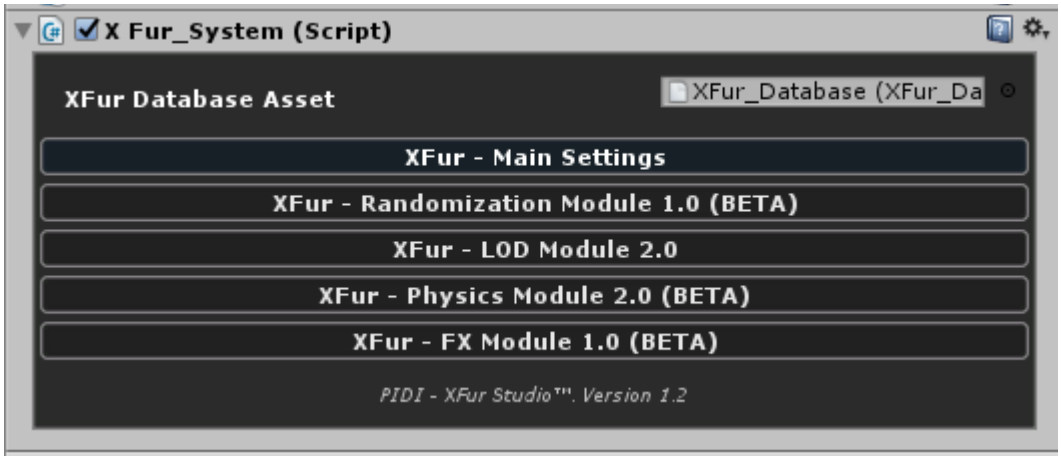


Fig. 14 – XFur System component up and running

Open the XFur Main Setting tab. You will see that the triplanar projection is working fine now and that your model looks much better.

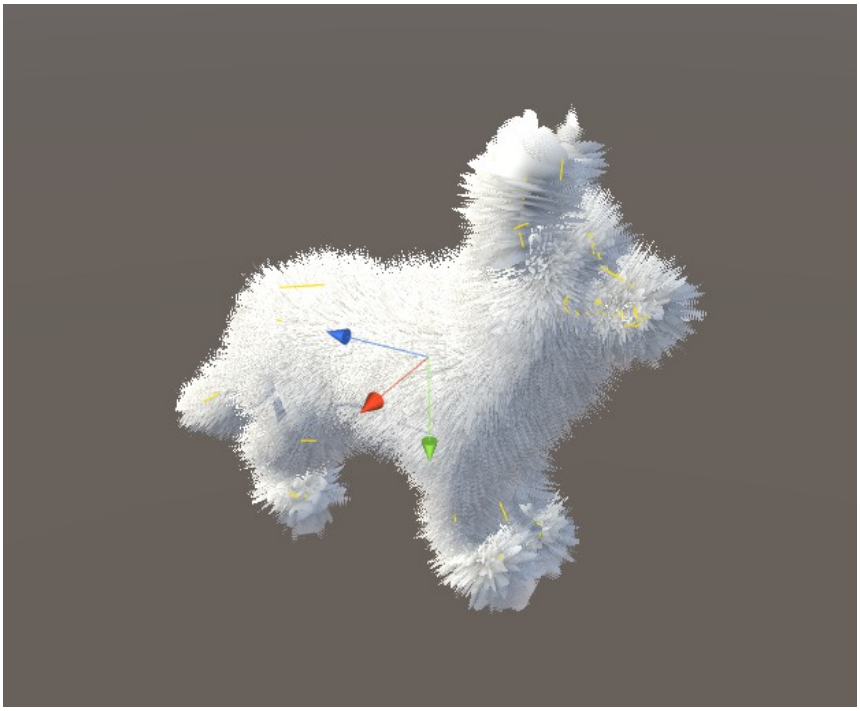


Fig. 15 – The model covered in fur after being patched by XFur System

With the XFur System component added correctly to your model, we can now get familiarized with every single part of the component.

V. XFur System. Main Settings

Fig. 16 – Main settings (basic view) in XFur System

This is the basic view of your XFur System's main settings. The Rebuild Data button forces XFur to re-generate all the patched meshes and re-calculate the fur. This is useful if you make changes to the original mesh, if you want to go back to a default state, etc. The Use Explicit Database Index mode allows you to assign a specific element from the XFur Database to this component regardless of the original mesh it had before. This mode should, in most cases, be used ONLY if you are dealing with procedural meshes or adding XFur to your models at runtime through scripting.

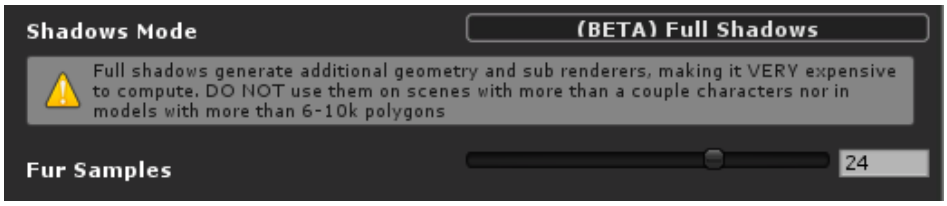
The Update Dynamically toggle should be enabled if you plan to modify the fur at runtime and both the FX and Physics module are disabled. Since it causes some performance impact, if you are sure that your fur won't be modified at runtime you can safely disable it to get a performance boost.

The material slot indicates which material you are currently viewing and editing. The main settings can be different for each material. The module's behavior is shared across all fur materials on this instance.

The Base Fur Material variable is the original material from which XFur will generate variations for LOD, effects, etc. If you want to change the materials of your character you need to change them here, in the Base Fur Material slot, rather than in the Materials tab of the Skinned Mesh Renderer or the Mesh Renderer.

Fur Samples is the amount of layers of fur that you want in your model. More Samples will give better results but it will also have a heavier performance cost.

The shadows mode switches between the Standard Shadows, which are fast to render and extremely performant though not accurate (they don't display the fur strands) nor compatible with forward rendering and Full Shadows, which are forward compatible and highly accurate but much more expensive to render.



When enabling Full Shadows, you can control the quality of the fur (the amount of samples) with a slider in a dynamic way. This is a functional improvement over previous iterations of this shadowing system and allows its usage on characters with higher polygon counts, with a slightly higher performance cost. This system is in BETA as of v 1.5.

The Basic Self Collision toggle tells the shader to use as simple algorithm to prevent the fur from going inside the model when using physics, wind or grooming. This is not a perfect solution as it is fairly basic for performance reasons but it will still prevent most issues with intersections.

The Triplanar Mode toggle overrides the material properties and can be toggled on and off for this specific instance. If Triplanar mode is enabled, the triplanar scale can also be overridden directly in the component.

The Force Grooming on UV2 toggle forces all grooming to use the secondary UV instead of the main one.

Next to this you can see three Popups for Base Skin Mode, Fur Settings Mode and Fur Noise Map. These popup let you decide from where will this instance read its fur settings, the original material or from the instance itself.

Finally the Load Fur Profile slot let's you assign a Fur Profile asset with pre-defined fur settings which will then be copied and assigned to this instance while the Export Fur Settings button creates a Fur Profile Asset with the current fur settings of this instance for later use.

We want to control the settings of the fur of this model through the XFur System component, so let's switch the Base Skin Mode, Fur settings mode and Fur Noise Map from “From Material” to “From Instance”.

Two new blocks of settings have appeared for us, a Base Skin and a Fur Settings block full of parameters we can customize.

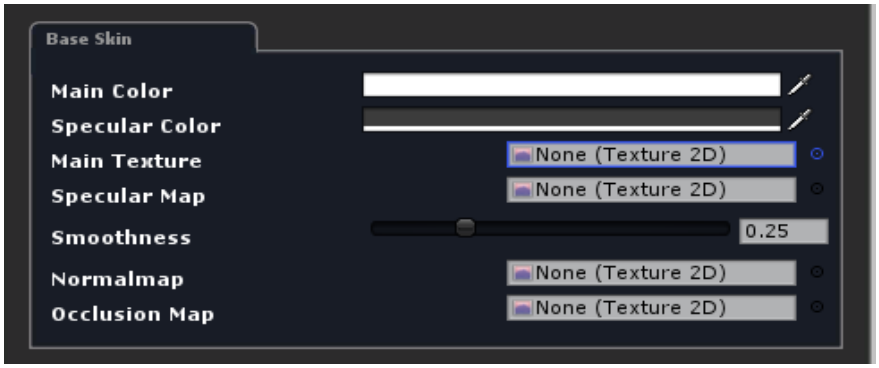


Fig. 17 – Base Skin settings as overridden by the XFur System Component

These blocks work exactly the same as the Material parameters, with the same names and functions. They act basically as a material override that let's you make per-instance variations without requiring new materials for every single character, color and pattern you need.

If you have a fur material for, let's say, a dog, you can share it across all breeds of dogs and customize it for each breed on the XFur System instead of having dozens and dozens of fur materials across your project. Or if you want to allow users to fully customize their characters without altering the main materials or creating runtime copies, the XFur System component makes this possible as well.

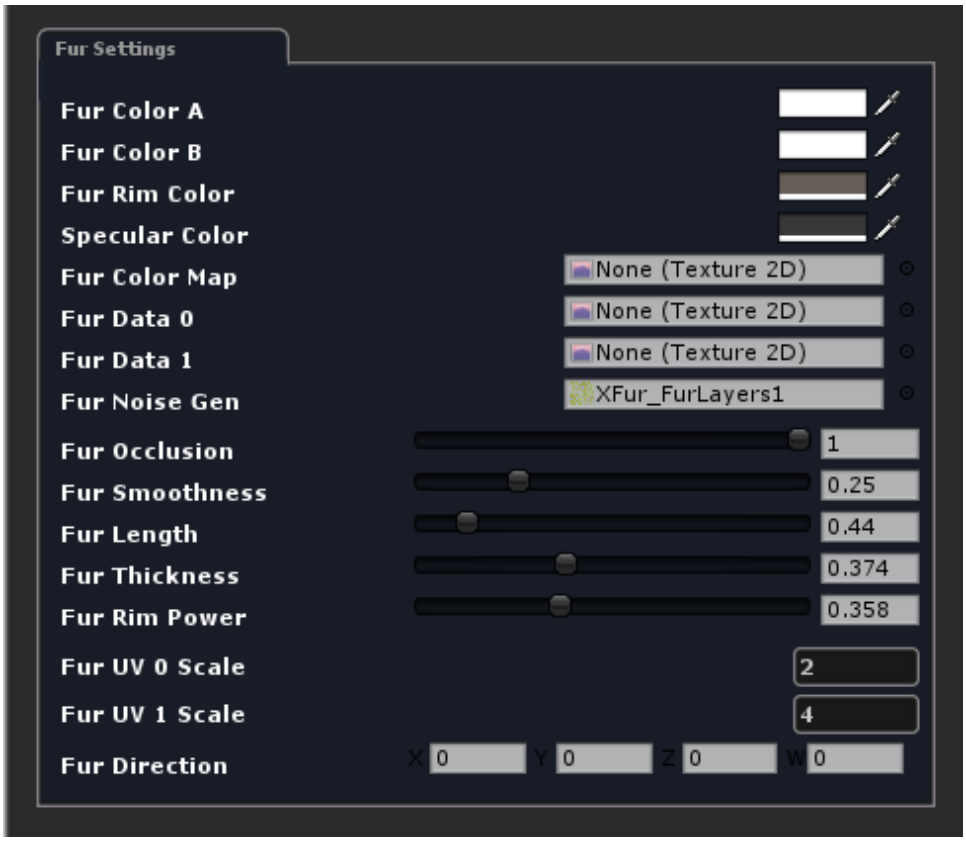


Fig. 18 – Fur Settings as overridden by the XFur System Component

Assign all the textures that you want on the Fur Settings and Base Skin blocks, adjust the length,

thickness, occlusion and other parameters of your fur as you wish, and once you are happy with how your fur looks, click on the Export Profile button and save the resulting asset wherever you want. Then, move to the next part of this manual.

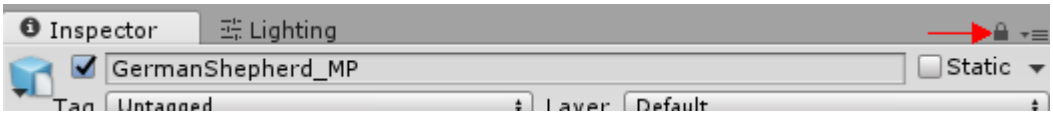


Fig. 19 – How to lock the inspector

In older Unity versions the Wireframe of the model may make it difficult to see how the fur looks as you edit it. To solve this, simply click on the lock icon over the inspector with the XFur System component visible and then deselect your model. Once finished, click on the lock again.

VI. Fine tuning fur with XFur Painter

Fig. 20 – Basic fur setup on a custom model, right before sending it to XFur Painter

Right now your model will not have the perfect look you need. You will most likely need to shave off fur from some areas, adjust the length of it on several parts, maybe even groom it.

XFur Painter, and additional tool provided with XFur Studio, is what we will use for fine tunning the fur and generating the required Fur Data 0 and Fur Data 1 maps (fur parameters and fur grooming respectively).

To start with XFur Painter, we will need to open the XFur Painter scene and drag our model there. There are three ways to do this : to create a prefab of the model we are editing right now, to copy it and then paste it on the XFur Painter scene or to add the model again in XFur Painter, assign the XFur System

component again and then load the profile we exported in the previous section by assigning it to the Load Fur Profile slot.

It is recommended to create a prefab and that is what we will do in this tutorial. Once your prefab is created, you will notice that the model's fur looks completely different since it is not applying any of the settings we defined. Do not worry about this, it is the intended behavior as XFur does not work when the assets are in prefab mode, it applies the changes only after they are placed on a scene.

Now that we have created our prefab object we will go to XFur Painter. Go to the Standard Pipeline subfolder of the package, and inside of it go to the XFur Painter 2.0 folder. There you will find a scene called XFur Painter. Open it.

Disable the Lucy_HighPoly game object provided as a demo and drag your model prefab instead.

Select the Xfur_Painter object in the Hierarchy view and you will see the basic settings for XFur Painter

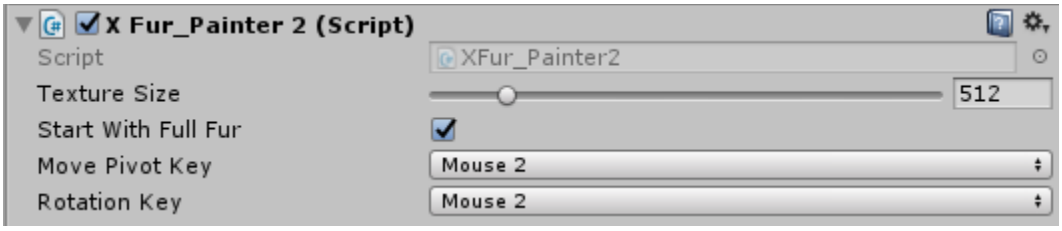


Fig. 21 – Default XFur Painter settings

From here you can set the texture size, whether you want the model to start completely covered in fur or not, which key to use to move the pivot when holding the Left Ctrl key and which key to use to rotate the view.

Select your model, where the XFur System component is located. When we paint the model, we will also paint the sections that we want to be affected by fur and physics and which ones we want to stay stiff.

In this dog model for example, we will want the ears, snout and feet of the dog to not be affected by physics nor any other forces since the fur is very short and it could easily go inside the geometry.

To enable the wind response, we will open the XFur – FXModule 1.0 tab and click on the Enable Module Functions toggle.

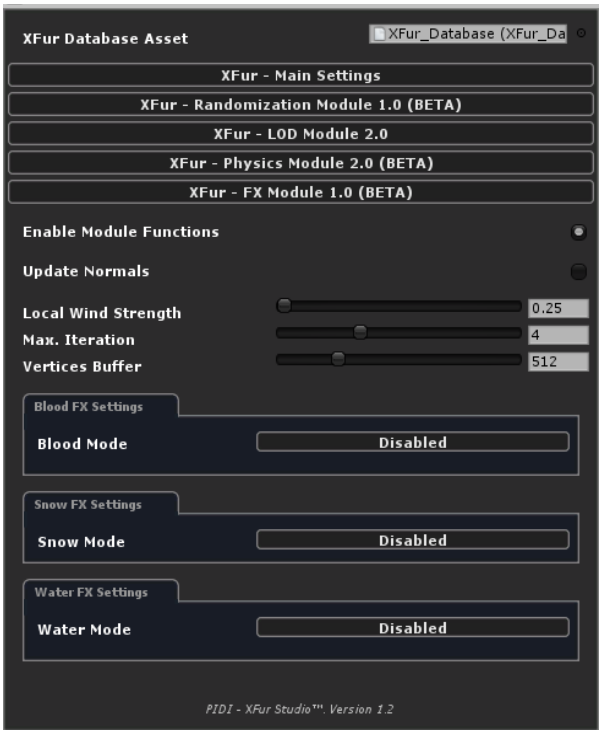


Fig. 22 – Enabling wind effects on the FX Module

There are many settings in this module but they don't interest us right now. Right now, we only need to set the Local Wind Strength to any value higher than 0. We will use 0.25f right now. If your model is very small you may need a higher value. Since we will be grooming our model, we also need to change back the Fur Direction vector in the Main settings tab to 0.2, 0.2, 0.2, 0.2.

NOTE : For better results when grooming, ensure that the root bone specified in the SkinnedMeshRenderer component is a bone whose orientation matches the standard (Y Up) Unity orientation. If not possible, assign the root object of the whole character's hierarchy as the root bone object in the SkinnedMeshRenderer component. This will require you to re-define the bounds by hand but will not change in any way the animations. This simple step will ensure that the grooming is affected by the rotation and movement of the character's bones.

If everything is in order, hit the play button on the Unity Editor.

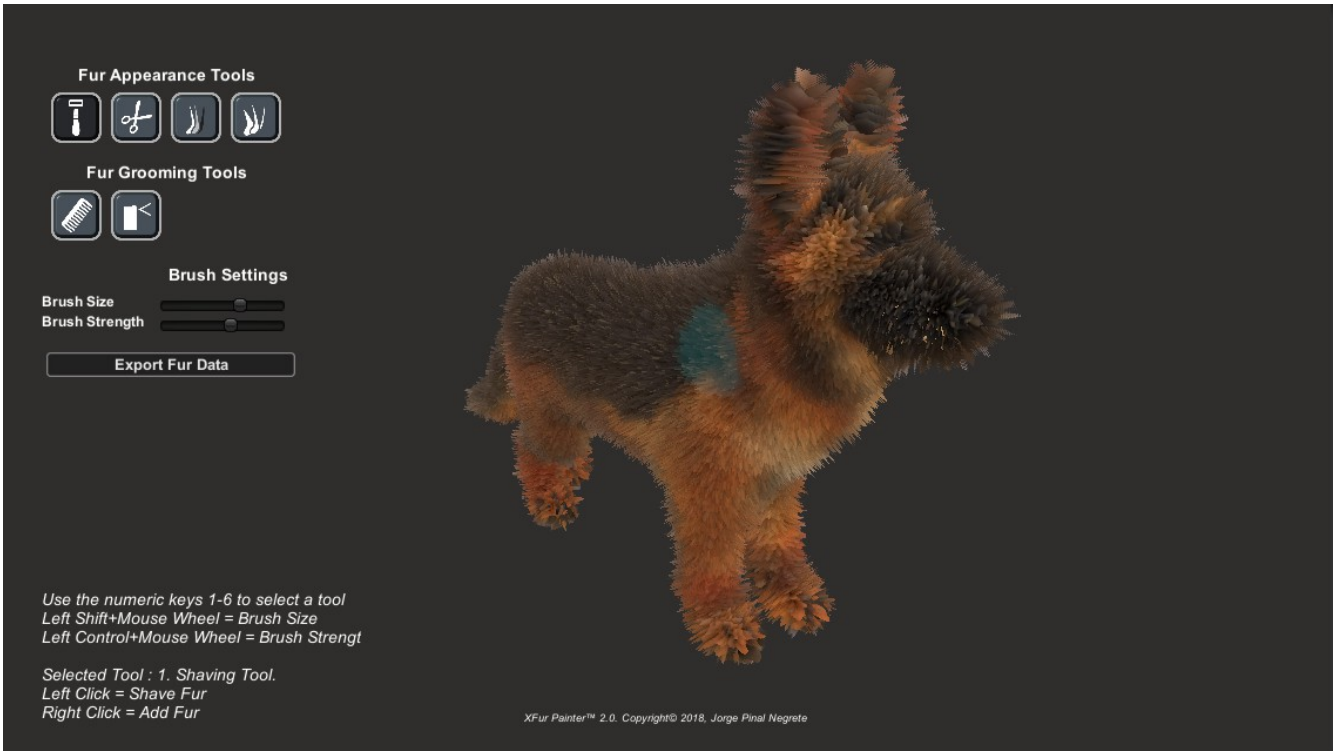


Fig. 23 – XFur Painter- Main view

This is the main view of XFur Painter 2. On the left side you have a panel with all the required tools to customize and groom the fur.

The Shaving tool allows you to add and remove fur from your model. With left click you can add fur over any section of the mesh and with right click you remove it, as simple as that.

The Trimming tool lets you adjust the length of the fur with high precision. Left click will make the fur longer (up to the maximum length you set up in XFur System or the XFur material) while right click will make it shorter. To control the speed with which you trim the fur you can modify the brush strength with the sliders or by pressing Left Control and moving the mouse wheel.

The Shadowing tool lets you control the strength of the self-shadowing applied to the fur. With left click you add stronger shadows and with the right click you remove all shadowing.

The Thinning tool lets you control how thick or thin will the fur look. With left click you make it thinner, with right click it will become thicker.

The Grooming tool lets you control the direction of the fur by dragging your mouse over the model's surface. With the brush strength controller you can regulate how strong will be the grooming force applied to the fur.

The Stiffening tool lets you control how rigid or loose will the fur look when affected by wind, gravity, physics etc. Left click makes it more stiff, right click removes stiffness.

Finally, the brush settings let you control the size and strength of the brush easily and without any issues.

XFur Painter does not provide perfect accuracy since it depends on mesh colliders and these have some limitations in Unity. The best way to have pixel perfect fur masks and parameters is to author these in a graphics software such as Photoshop or GIMP. However, for most cases, XFur Painter is a great tool to do the fine tuning of your furry creations.

Once you have finished getting the right look for your fur, you can click the Export Fur Data button. The resulting textures will add the material name and the data type name at the end of the name you choose for the file, so it is recommended to use a generic naming such as "Dog_FurMasks since the software will add, on its own, more information to these names. The resulting name of the textures for this example would be "Dog_FurMasks_MaterialName_Mask.png" and "Dog_FurMasks_MaterialName_Groom.png".

These textures will be saved and imported into your project with the appropriate importation settings and parameters. They are ready to use right away, and if you assign them to the XFur System component in the corresponding slots and go back to XFur Painter they will be loaded automatically so you can continue your work from wherever you left off, making painting and customizing your fur easier than ever before.

If you wish to author your own Fur Data Maps without the tool (mostly to get a higher resolution result or even more precision) the channel configuration is the following for the main fur data map : Red channel = Fur mask, Green channel = Fur length, Blue channel = Fur shadowing/occlusion, Alpha channel = Fur thickness.

For the fur grooming map, any value above 0.5 adds positive displacement of the fur on the X, Y or Z axes (red, green and blue channels) while any value below 0.5 adds negative displacement. The alpha channel is used for fur stiffness with a fully transparent Alpha channel meaning full stiffness.

3. XFur Studio : Advanced Modules

I. Randomization Module 1.8 (BETA)

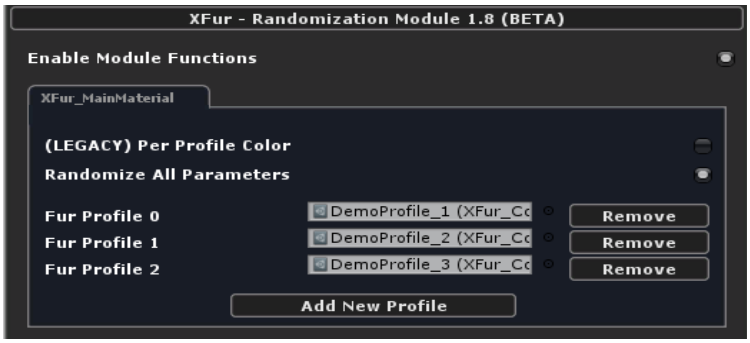


Fig. 24– The randomization module UI

The Randomization module lets you assign any number of fur profiles that you have previously exported and will pick one at random on startup or creation for the case of prefabs. This way you can have many variations of the same instance when creating packs of wolves, herds of cows or any other similar use.

This module is currently in Beta and under heavy work to add new features. Such features include random coloring and fur patterns to truly produce endless variations even with just a couple of profiles.

To use the Randomization module simple open the Randomization Module Tab on the XFur System interface and then click on the Add New Profile button to generate the empty slots where you can drop your profiles.

II. LOD Module 2.0

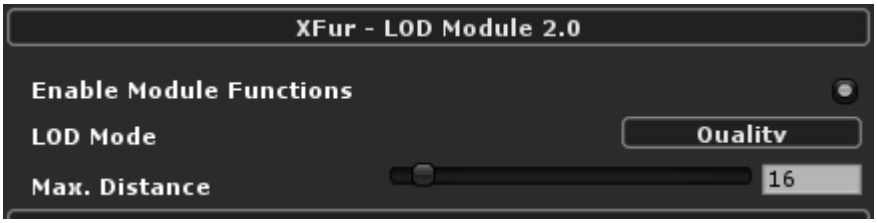


Fig. 25– The LOD Module's UI

The LOD Module contains a very easy to use algorithm that switches between the different rendering levels to simplify the rendering of the fur (as well as to limit the calculations from other modules such as the physics one) to improve performance when the objects are far away.

It has two parameters, the LOD Mode and the Max. Distance. As the character nears the Maximum Distance the fur samples and the precision of other modules will be reduced. The LOD Mode controls the bias towards Quality or Performance.

XFUR is NOT fully compatible with the standard LOD component from Unity.

III. Physics Module 2.0 (BETA)

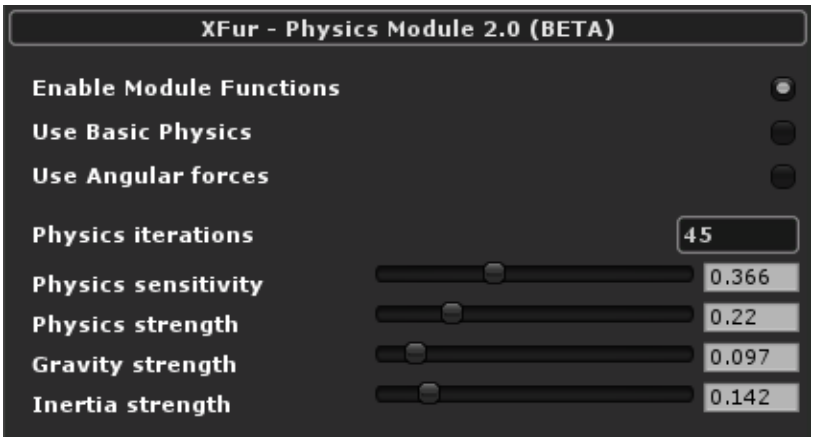


Fig. 26– The Physics Module's UI

The physics module simulates physics over the character using the XFur System. It automatically detects and handles all bones from the rig if the mesh is animated without needing any manual setup. Rigs of up to 128 bones total (including root and other non-weighted handles) are supported in this version.

The physics simulation is calculated using two different algorithms and two different sets of data depending on the balance between performance and quality that you want. The Basic Physics method reduces the amount of operations to calculate the displacement and rotation of each bone or the object itself at the cost of some precision in the results.

Angular Forces refer to the impulse and physics derived from the bones rotation. While these add more realism in some cases they also duplicate the amount of operations needed for the physics simulation and are not worth the extra cost in a lot of cases.

The Physics iterations value determines how many times will the physics be updated and sent to the fur. The higher the value, the smoother the results will be, but the heavier they will be to compute.

The physics sensitivity value determines how sudden or big should the movements be before they start registering in the physics simulation. If the sensitivity is too high, something as subtle as the character closing his fist will produce physics dynamics.

The gravity strength controls the global gravity applied to the fur and the inertia strength adds some basic inertia/spring to the whole physics simulation. In most cases, it can be set to 0 without noticeable effects.

Starting with XFur 1.8 you can manually define which bones will be tracked by the physics module by disabling the Auto-track bones toggle (available only on skinned meshes). Once you have defined which bones will be tracked or if you make any changes to these bones, you will need to re-build the mesh data to ensure all changes are applied.

IV. FX Module 1.0 (BETA)

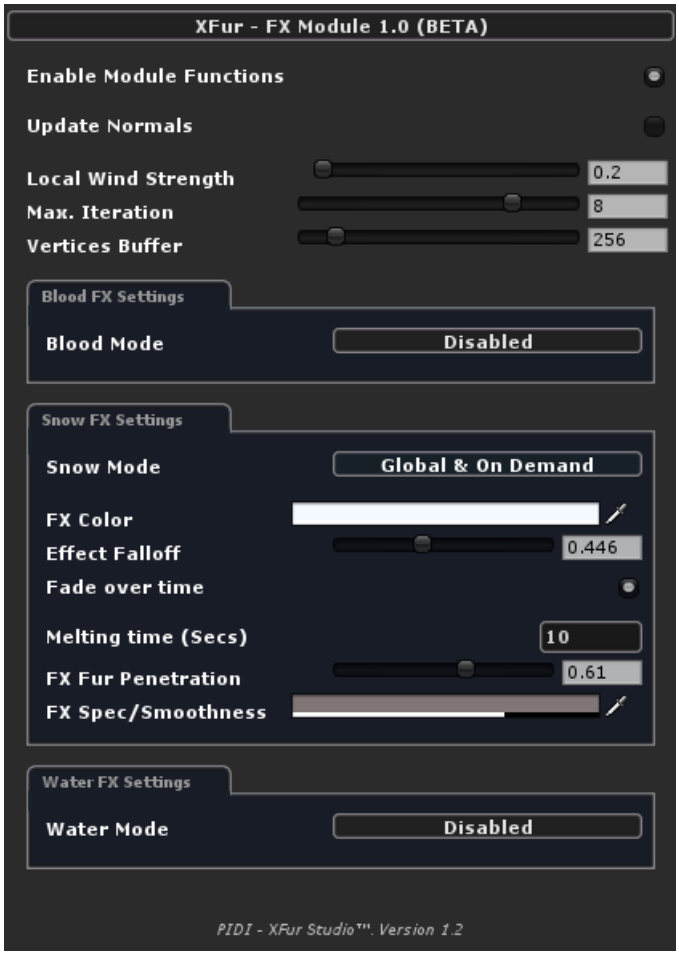


Fig. 26– The FX Module's main UI

The FX module controls the interaction of the fur with the global weather as set up in the XFur_WeatherSystem component. These two components go hand in hand and to successfully use the FX module you need to make sure to have one object in your scene with the XFur_WeatherSystem component attached.

The Update Normals toggle is used for animated meshes and makes sure that every time the FX module updates it also updates the normals of the animated mesh to make sure that snow and water coverage are affected by the animation cycles and positions.

The Local Wind Strength overrides the one specified on the material and controls the intensity of the wind for this specific instance.

The Max. Iterations controls the maximum amount of updates that the FX module can have in a second. Since this module operates on a vertex-basis this number is limited to a low iteration count.

The vertices buffer determines the maximum amount of vertices that will be processed in a single frame. The lower the amount of vertices the choppier the effects' transitions will look but the higher the performance you will get. A higher vertices buffer size will produce much smoother results at the cost of a performance drop. If you have few characters in your scene, you can safely increase the vertices buffer.

In this first version of the FX module there are three supported effects : Blood, Snow an Water.

Fig. 27– Specific FX parameters (snow)

When you enable an effect its block of settings is displayed on the UI. The Effect Mode defines how it will be updated. Global & On Demand will update the effect both when it is assigned directly with the function ApplyEffect(int effectIndex, int[] vertexIndices, float intensity = 1) and with the values from the WeatherSystem.

Global Mode will update it only through the Weather System and On Demand will update it only when called from the ApplyEffect function.

Snow and Water can be set to work with the weather system, blood can be used only on Demand mode. While the only way to apply blood to the fur right now is to provide the vertex indices list through code in future versions there will be additional tools that will automate this process when coupled with colliders, rigidbodies and particle systems.

The FX Color is the main color the effect will have The Effect falloff determines, when using the Global mode, how the effect will cover the areas of the mesh that are not pointing towards the direction in which the rain/snow are falling. It is, in other words, a slope controller.

If Fade Over Time is enabled, the effects will fade out naturally as time goes by and the Melting/Fading time allows you to set up how long in seconds this fading out will take. The lower the value the faster the snow and other effects will melt and the higher the intensity they have will need to be for them to stick to the fur's surface.

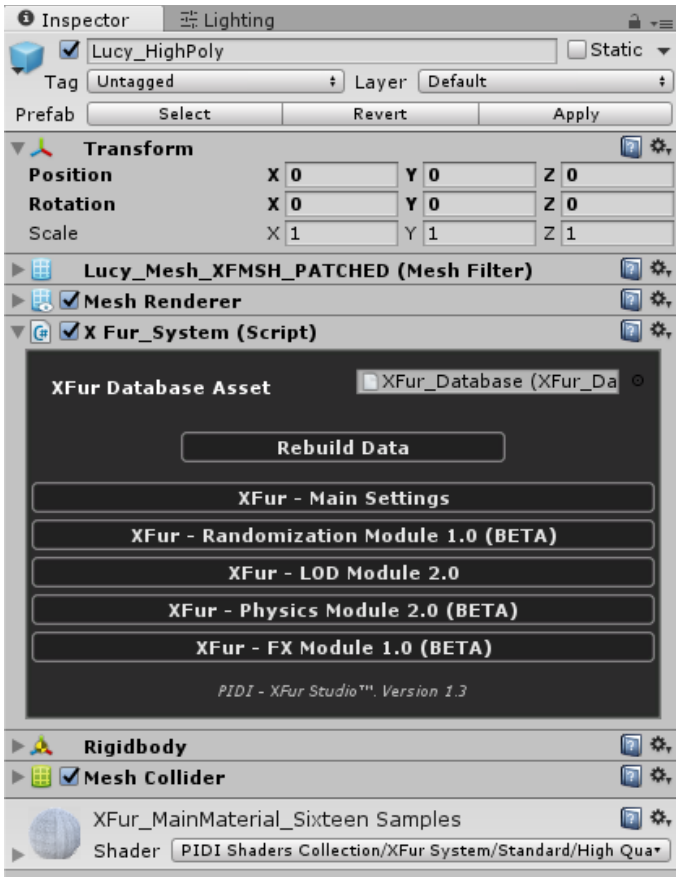
The FX Fur Penetration controls how far into the fur the effect will get so it can cover only the tips of the fur or up to the very roots of it. In the case of snow and water, this also produces fur clumps.

Finally, the FX Spec/Smoothness color controls the specularity (RGB) and smoothness (Alpha) of the effect over the fur so the water effects can properly make the fur appear "wet".

Internally, the effects also modify the way the fur reacts to physics, wind and gravity as well as its length and general appearance for a more realistic result.

Starting with version 1.5, XFur Studio includes three new components that allow you to apply effects over the fur of creatures that either collide with another object, particles or that enter a certain volume. These components are fully configurable and can suit most needs but they also serve as a great starting point for you to code your own interactions and applications for FX over XFur characters.

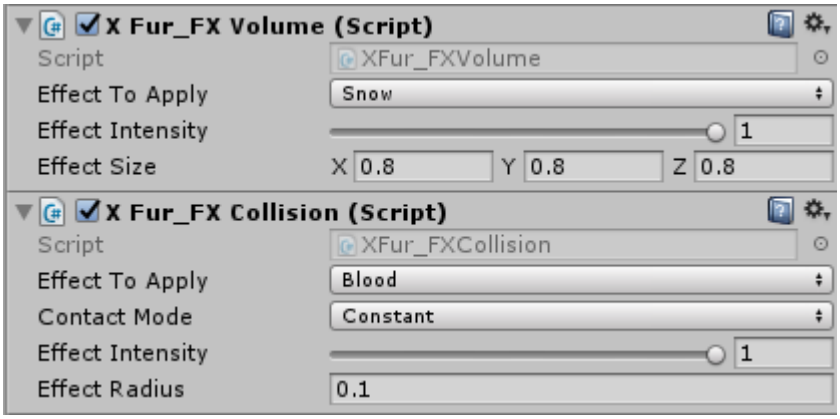
Using these new components is very easy :



First, make sure that your character has a XFur System component correctly attached, as well as a valid collider (it doesn't need to be a Mesh Collider) and a Rigidbody components.

In the XFur System component, make sure that any effects you want to apply through collisions or volumes has been set to Global&On Demand or On Demand mode :

Then, either on an empty game object or in the object you want to act as a collision detector, add the FX Volume or FX Collision component. Both work in a very similar way :



The XFur FX Volume component allows you to detect collisions and presence inside of a box-shaped volume. The effects are applied in a constant rate for as long as the fur covered characters are inside of its bounds. You can select exactly which effect to apply and its intensity.

With the XFur FX Collision component you can simulate effects being applied upon collision with a spherical collider. In this case, you can also decide if the effect should be applied constantly (as long as there is a collision) or only upon first contact.

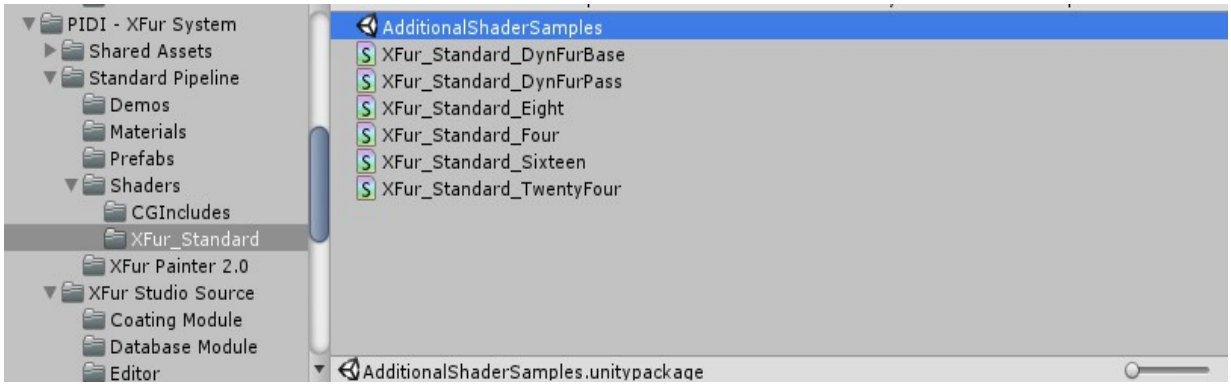
These components when used together with the right settings on the XFur System component for your characters will allow you to simulate very complex interactions between the fur and the environment or other characters, be it to simulate complex damage from wounds and shots received in real-time to a snowball correctly hitting a character or a bucket of water wetting only certain sections of the fur.

4. Additional Shader variants

Starting with XFur Studio 1.4 there are multiple new variants of the standard shaders designed to provide new sample amounts and better transitions both for the LOD module as well as more flexibility when working on both lower and higher qualities.

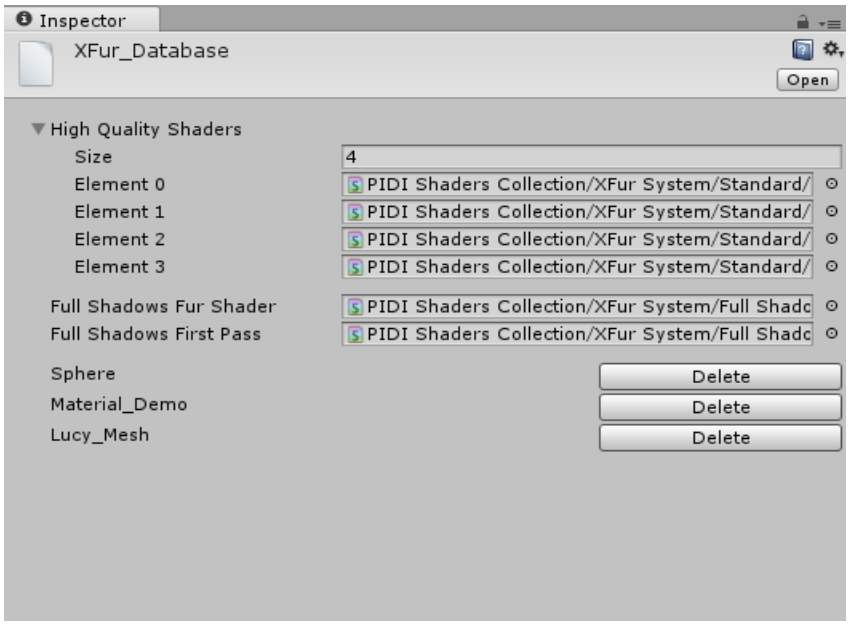
Shader variants for 12 and 20 samples have been added to cover smoother transitions between fur qualities as well as 28, 32, 36 sample variants to allow for higher qualities than in previous versions of XFur.

These new variants are, however, packed inside a UnityPackage file to avoid unnecessarily long compilation times in projects that won't use them. If you need them for your project you can add them as follows :



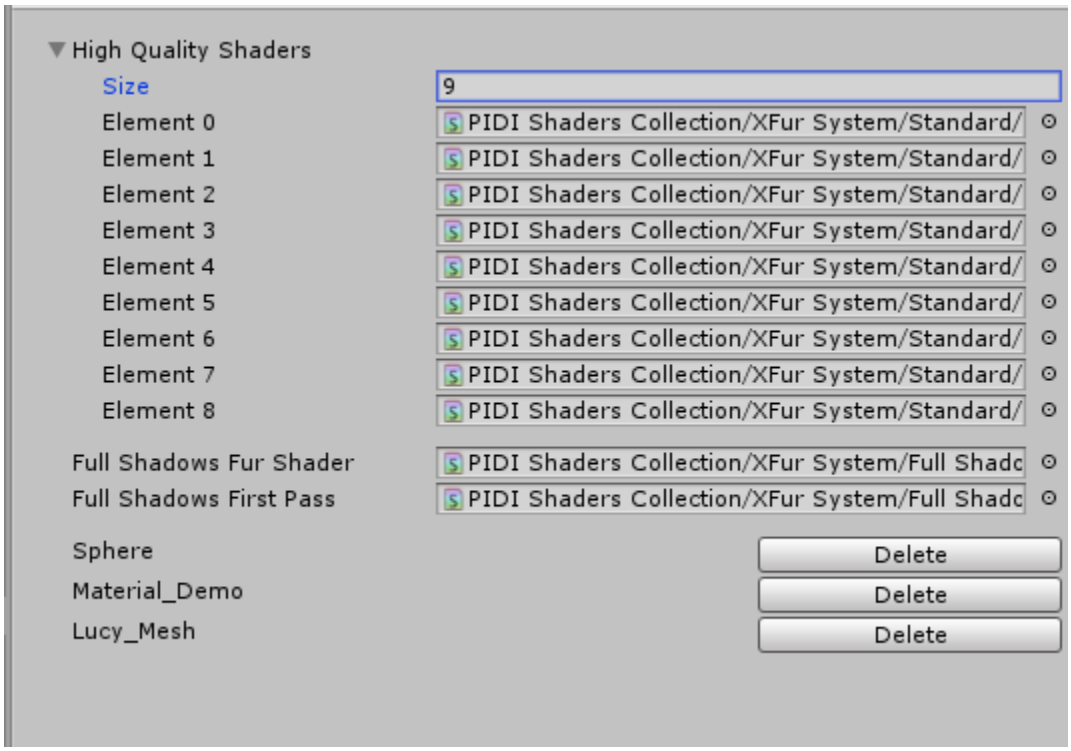
Unpack the shaders included in the AdditionalShaderSamples package inside the StandardPipeline/Shaders/XFur_Standard folder.

Then, once all the new shaders are extracted and finish compiling, create a new empty scene or go to a scene without any XFur enabled object and select the main XFur Database(s) used on your project.



Select the High Quality shader's array and resize it to have enough space to add as many of the additional variants as you need. Then, inside the array, assign them in such a way that they go from the lower sample amount to the higher sample amount.

In the default database, the shaders are assigned in order, from top to bottom, as 4,8,16 and 24 sample shaders. If you were to add all the additional shaders to the database it should have the shaders assigned in the following order : 4, 8, 12, 16, 20, 24, 28, 32 and 26 samples.



Then, when you re-open a scene with your XFur enabled characters which have this database assigned to their XFur System component, their available LOD options will be automatically updated to show all the new available variants without requiring any additional work on your part.

Remember that these shaders are not AUTOMATICALLY UPDATED. You need to unpack them again with every new update so that the newer versions can replace the ones that you unpacked before, overwriting them and adding any new features that may be missing.

5. XFur Studio & Scripting

XFur Studio is intended to be used through the different UIs and tools to make the whole workflow as easy as possible. However, for advanced users and very specific situations you have many functions and interfaces available, besides full access to all the components and systems.

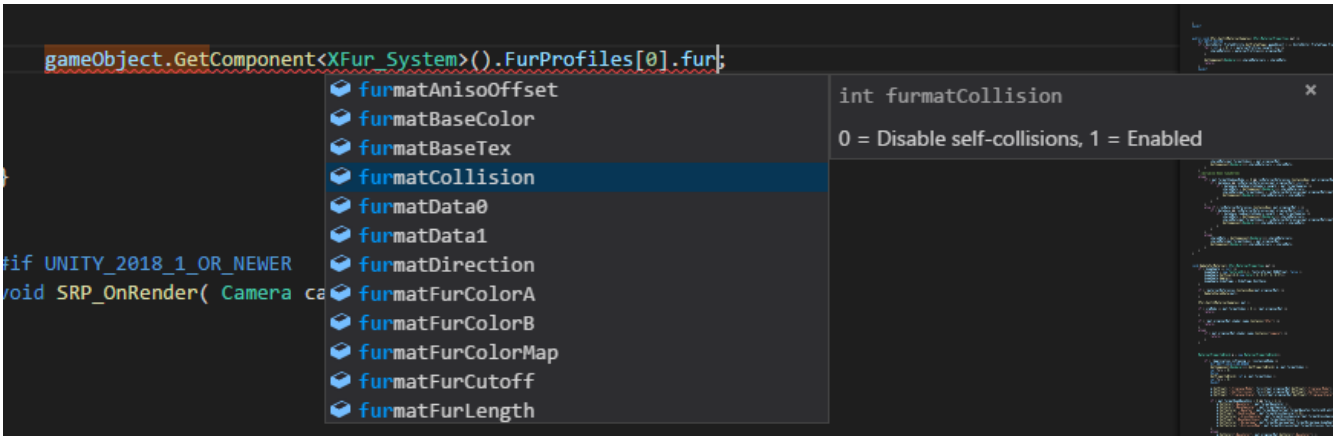
All XFur classes are contained within the XFurStudio namespace. To call any function from the XFur system or any of its components, you must write **using XFurStudio** at the top of your C# script.

The main entry point for any function is the XFur System Component. Usually, in your scripts, you would access it like this :

```
gameObject.GetComponent<XFur_System>();
```

Where gameObject is the target object from which you want to access the XFur_System component.

To access the fur properties of any of the materials, you use the FurProfiles interface and the index of the fur material you want to edit. After this, if you have auto-complete enabled on your coding IDE, you should see all the available methods and variables with a small explanation of what they do :



Any function that does not have a description should not be called by your scripts, as they are most surely intended for internal use only. Do not call nor use any internal function unless you are sure of what you are doing.

The main functions that you can call from other scripts and that you may need at some point in the development of your project are the following :

In XFur_System :

public void LoadXFurProfileAsset(XFur_CoatingProfile profile, int materialIndex)

This function lets you assign a fur profile (**profile)** to the XFur system manually, to the specified fur material (**material Index)**. This is useful if, for example, you are allowing your players to customize their characters and need to let them choose among a series of fur patterns. Simply assign the profile that corresponds to the choice of the player, and load it with this function to make the system load all the fur properties.

public void ApplyFurProperties(int profileIndex = -1)

Forces all materials (or the specific material with index equals to profileIndex) to update and apply any changes that you have made.

WARNING : Once an XFur System component is added to an object, it will override and take control over the Materials array of said object's renderer. If you want to swap the materials of the object at runtime through code you need to override both the original material from the corresponding XFur material profile (this step is optional on Unity 2018.3+) and the material itself on the sharedMaterials array of the renderer :

```
int materialIndex = 0;//The index of the material we want to swap.
Material newMaterial; //The new material we want to assign

gameObject.GetComponent<XFur_System>().materialProfiles[materialIndex].originalMat = newMaterial;
var sharedMats = GetComponent<Renderer>().sharedMaterials;
sharedMats[materialIndex] = newMaterial;
GetComponent<Renderer>().sharedMaterials = sharedMats;
```

Do not swap a XFur material for another XFur material. Use Fur Profile Assets instead.

In the XFur_FXModule class :

public void ApplyEffect(int effectIndex, int[] vertexIndices, float intensity = 1)

Applies the specified effect (0=blood,1=snow,2=water) to the specified vertices with the given intensity. Changes will be applied in the next update call of the module.

While most other functions are also available and public to access from other scripts, you should avoid calling them directly as it may cause conflicts with the behavior of the system since most of it is automated.

You should not call any of the functions started with **PDEditor** since they are designed to be used inside the Unity Editor only, to design and present the different UIs and custom inspectors.

Adding XFur dynamically in v1.5+

Starting with XFur 1.5 we have added a new demo showcasing how to properly add XFur at runtime to a model as well as how to link the data of a mesh to an explicit index inside the database rather than automatically to an original mesh. This is useful mostly for cases where the meshes used for XFur are generated procedurally instead of imported as usual.

The proper way of adding XFur to a mesh at runtime is showcased on the new "Runtime" demo but it may vary depending on the way you are handling your meshes, if they are procedural or not, etc.

In general terms, this is the right process to add XFur at runtime :

- Add the XFur System component to an object that has either a combination of MeshFilter and MeshRenderer components or a Skinned Mesh Renderer. Either option should have already a mesh assigned to them, since this is required by XFur to work properly.
- Immediately after adding XFur to your object and still inside the same frame you need to assign a valid database to the newly added XFur System component.
- Call the Xfur_Start() function on the XFur System. This will launch the basic setup for XFur and prepare all the resources and startup values.
- Configure any and all settings you need for the XFur System component. If you don't want to set up all fur properties procedurally you can always load an XFur Profile dynamically from a previously saved asset.

6. Final notes and Performance Tips

XFur is a highly complex system for fur in Unity and as such it has not been designed to be used on low end PCs nor on mobile. To use it at its fullest a dedicated graphics card is heavily recommended.

The system has been tested on integrated IntelHD graphics cards and on GeForce GTX 6xx level cards. Dedicated graphics card had great results and high FPS counts even with dozens of 5-10k animated characters while integrated graphics cards had a moderate performance (20~30 FPS) in similar setups.

However, there are limits on the system. To get the maximum performance out of your scenes and characters we recommend you follow the tips below :

- Use deferred rendering whenever possible. Deferred rendering greatly reduces the amount of draw calls done by the shader. A 16 samples shader will add just 1 draw call for each additional point light after the first one is rendered in deferred mode.
- Do not light the fur enabled mesh with more than a few point lights plus a directional light in forward mode. If several lights are needed set the pixel light count to a low number.
- Do not affect a fur enabled model with many shadow casting lights. Each shadow casting light will re-draw the whole fur effect so a 16 samples shader affected by 2 shadow casting lights will make up to 54 draw calls even in deferred mode.
- Use the LOD module whenever possible. This module will dynamically adjust the number of samples based on the distance to the camera for maximum performance as well as the resources used by the other modules.
- Keep your polygon counts low. XFur shaders render the geometry multiple times and even generate their own geometry when using full shadows, so a polygon count between 1-12k is the best way to improve performance. These limits refer to the polygons actually covered in fur. If you have a 20k polygon character with fur only on his face then the performance considerations apply only to the face itself.

These tips are just a general advice and can be ignored. In more modern PCs the performance cost of most of these situations is not too high.

If you have any troubles with our software or need any assistance, don't hesitate in contacting us at support@irreverent-software.com and we will get back at you with help.

Thank you very much for purchasing our software, we hope our tools and the PIDI Game Development Framework will help you make awesome games!

Jorge Pinal, Irreverent Software™