editing and porting hair extra skeletons

guide by @sleepybnuuy 🐇💤

this brief guide will walk through the following:

- what is an extra skeleton, why do i want it, and how does it work?
- extracting a skeleton with VFXEdit
- editing an exported skeleton in Blender
- importing a finished skeleton through VFXEdit
- adjusting and applying the skeleton's physics
- adding brand new bones to a hairstyle for posing

none of this is exhaustive - it may in fact involve bad practices or janky spaghetti code! However it should work and shouldn't break your game.

This guide assumes you know some of the following:

- basics of importing and exporting mods through textools or penumbra
- fundamentals for editing mods through blender
- familiarity with weight painting or weight transferring
- use of posing tools like anamnesis, ktisis

if you're foggy on any of the above, this may not be the guide for you!
however, if you've ever edited a hair and wished for better/different weights and bones, this guide is probably for you 🐇

SPECIAL NOTE 6/28/25: i've made a followup to this guide that goes into a bit more detail on mashing multiple skeletons together – this one is primarily focused on porting a unique skeleton between races. you may benefit from reading one or both depending on your project, so the link to that is here: HOW TO USE MULTIPLE HAIR SKELETONS FOR 1 MASHUP

RECOMMENDED TOOLS include:

- Textools
- XIVLauncher / Dalamud
- VFXEdit
- Blender 4.0+

what is an extra skeleton?

extra skeletons are bonus sets of bones that XIV will load under certain conditions. If a weapon has a spinny thing on it, a chestpiece has dangly bits on it, or hair has extra parts to it, these can all be controlled by the addition of an extra skeleton to the playermodel.



These are super useful for a variety of reasons, such as adding custom animations to a weapon beyond just particle effects, or in the case of hair skeletons, giving the model more bones to be weighted to. Unlike the player skeleton, extra hair skeletons are controlled entirely by the game's physics engine - you don't have to touch a single animation file to make them move around, which is super nice for developers and also for us who like nice flowy hair.

You'll still have to weight paint or transfer weights involving these bonus bones, but being able to add as many as you like, or in as many positions as your hairstyle needs, is obviously a great benefit to have. Many hair mods tend to grab the best-fitting skeleton from an existing hairstyle on your race/gender and weight to those bones - but if you can define and edit your own extra skeleton, you can make a totally new set of bones to play with (and even give them physics! no more rigid ponytails)

how do they work? what's the gritty details

as with a lot of VFX modding, the answer may often be "We just don't know." however, at a high level, the following is true:

- Player skeletons are defined as **sklb** files, which control the layout of bones to be animated
    - These bones are animated with other VFX files (.vfx, .pap, .eid, etc)
- Extra hair skeletons are also **sklb** files, but are conditionally loaded and applied based on the hair's metadata
    - Bones defined by an extra skeleton are animated fully through **phyb** files, but can still be manipulated with gpose tools like Ktisis or Anamnesis
    - Extra Skeleton Parameters (**EST**) in an object's metadata can be configured either with Penumbra or Textools, signaling to the game to apply a particular skeleton ID when a gear piece or hairpiece is loaded
    - Hair models can be weighted to both player armature bones and extra skeleton bones, meaning that when the proper skeleton and bones are loaded, the extra skeleton and its motions/posing can affect the hair mesh

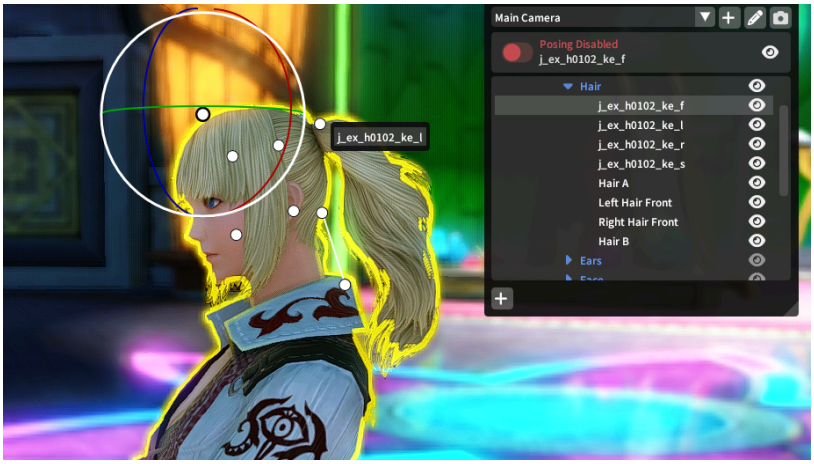
It is **far** easier and safer to muck around with extra skeletons than player skeletons - because they're loaded in as extra bits and never referred to on animations, there's no big risk of compatibility issues with other mods, body types, etc. Less likely to crash and burn too.

extracting a skeleton with VFXEdit

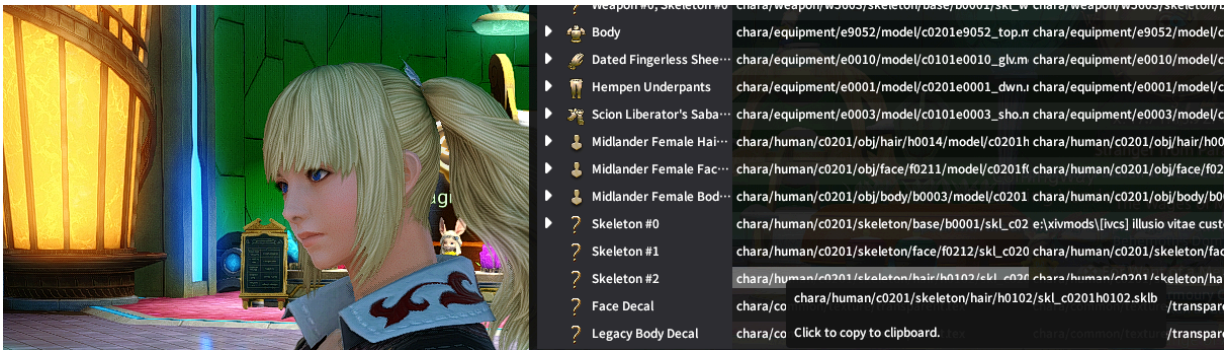
you should install VFXEdit through Dalamud and browse its documentation here: https://github.com/0ceal0t/Dalamud-VFXEditor

For an example project, we'll explore a workflow for porting MIDLANDER F HAIR 14 and its extra skeleton to midlander masc. This is a good candidate for a skeleton port, because the extra skeleton for this hairstyle belongs exclusively to F-midlander models - meaning if we wanted to weight something similar on a guy, it'd be exhausting to get the twintails fitting and moving naturally without a matching skeleton.

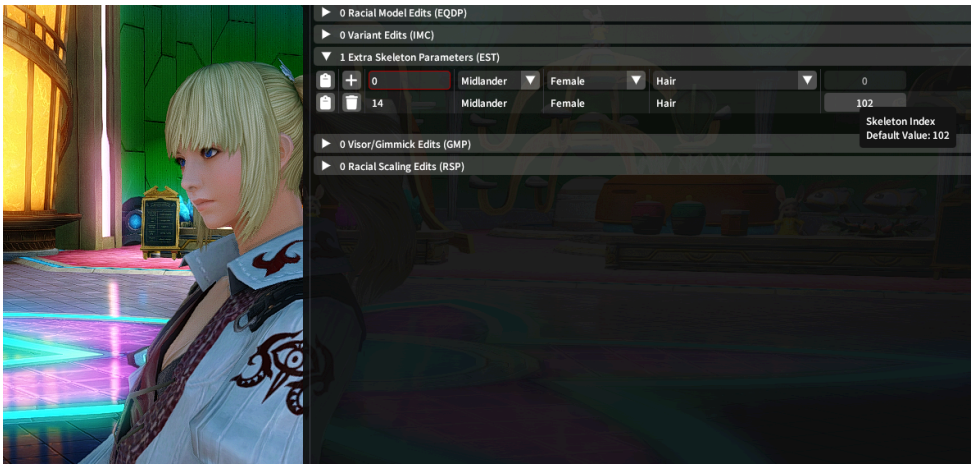
Once you've identified a hair that you'd like to edit the skeleton of, first identify the filepaths that said hairstyle uses for its skeleton and physics. You can observe these either through penumbra's on-screen viewer tab, the bones available in ktisis/anamnesis, or the metadata tags linked with your hairstyle:



Identifying extra skeleton bones via Ktisis

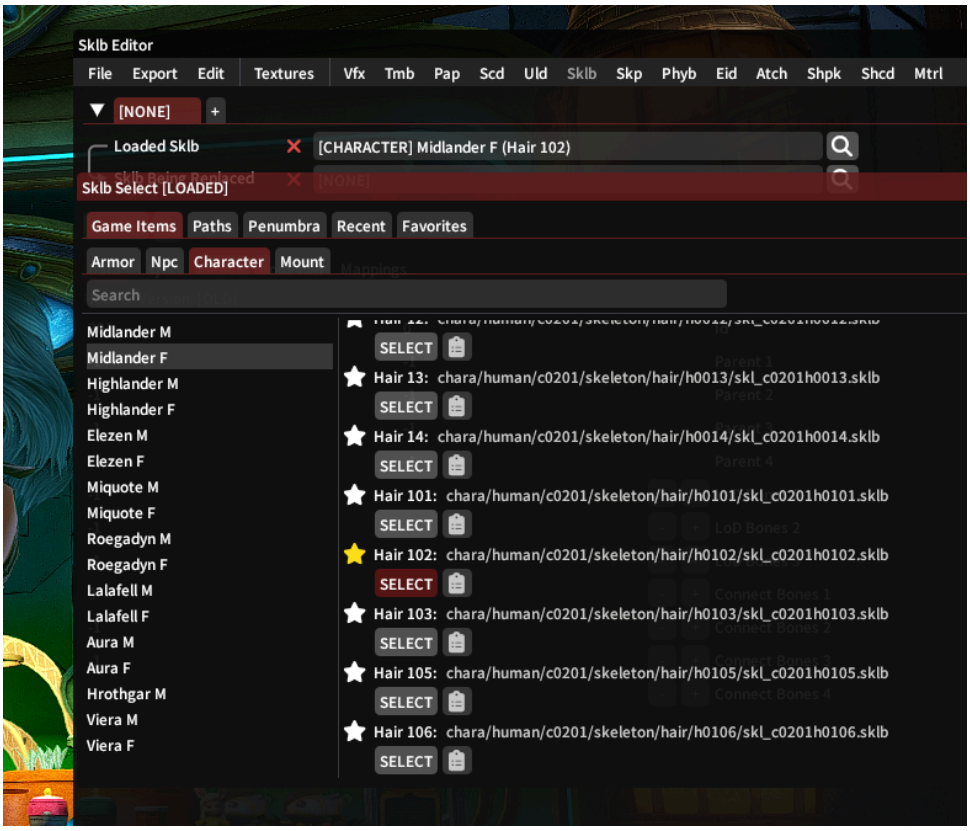


Identifying extra skeleton filepath in Penumbra

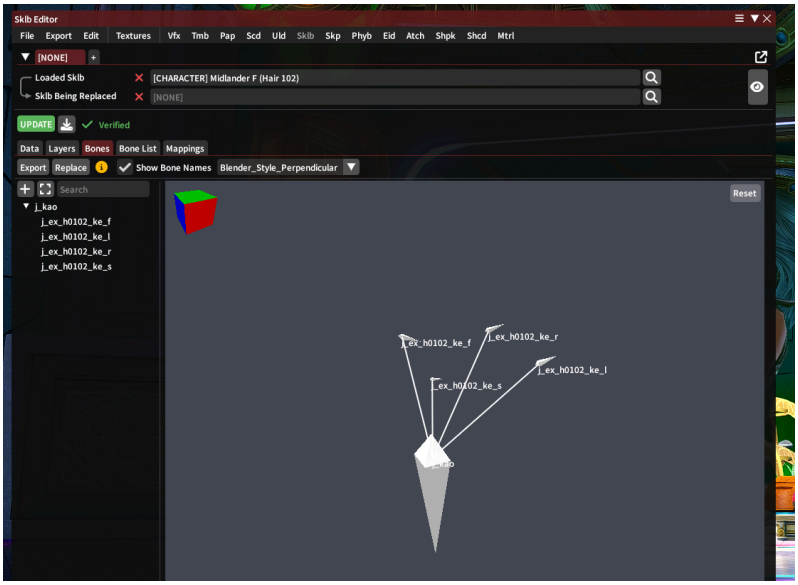


Identifying skeleton index with EST metadata

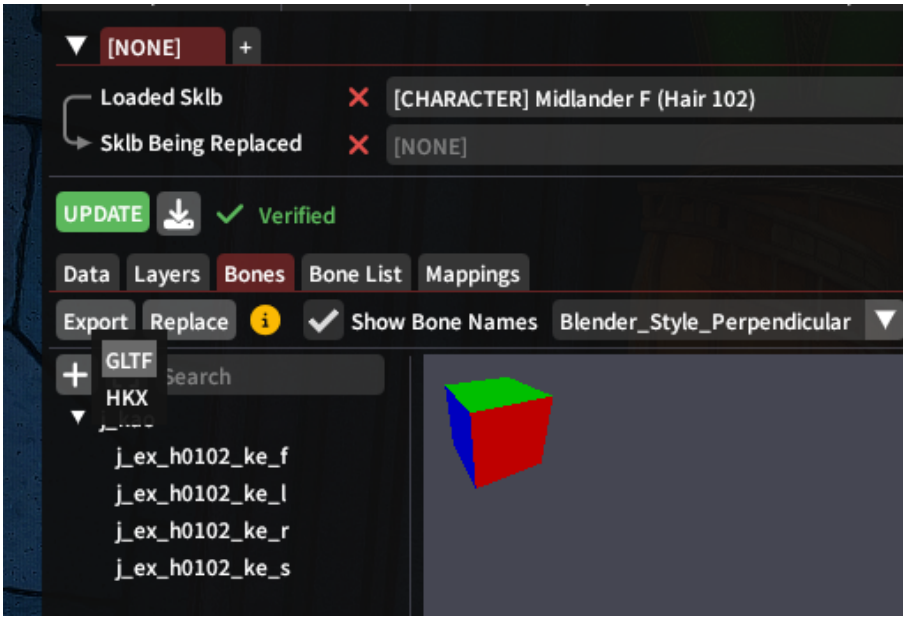
Once you've identified the skeleton to rip, we can use vfxedit to find and export the SKLB file.



In VFXEdit's SKLB panel, load your desired hair SKLB by browsing underneath the race and gender

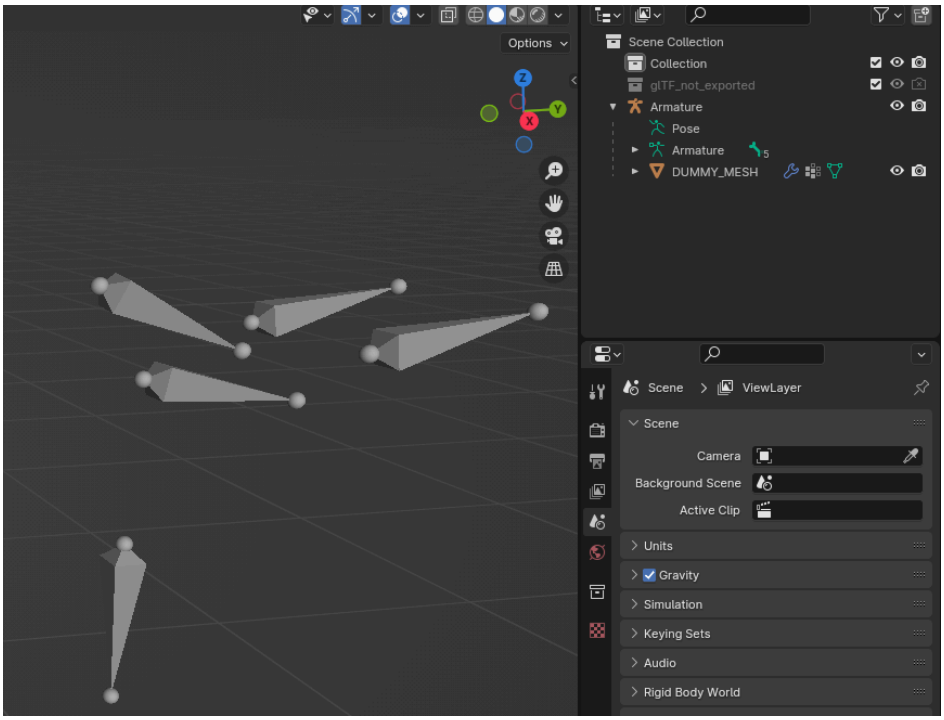


Wow, real bones just like in science class



Use the GLTF option from the export button to save this skeleton as an armature

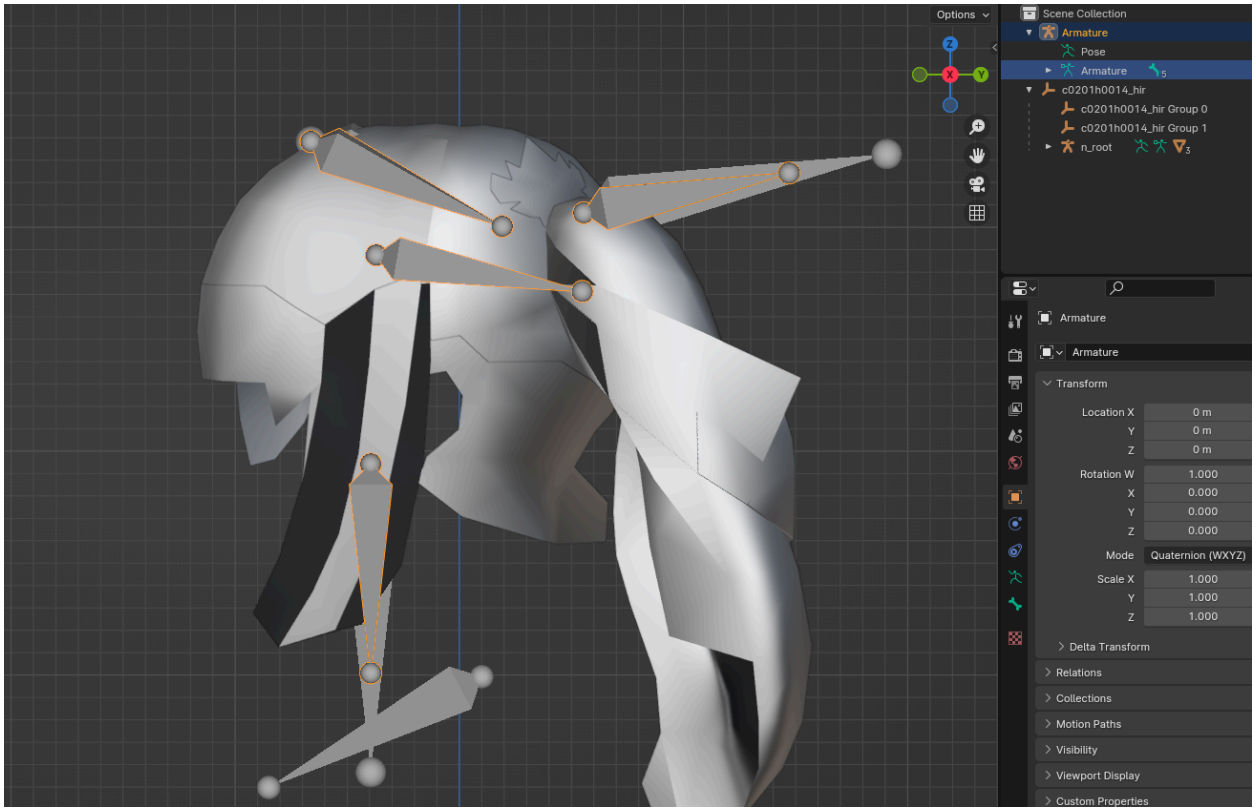
With your GLTF armature saved, you can import it to Blender 4.0+ (follow [these instructions] for import settings if you run into issues)



You can delete the Icosphere, DUMMY_MESH, and glTF_not_exported elements to make your armature look like normal bones

editing an exported skeleton

To verify that all our bones are correct, we can load in an exported FBX of F Midlander hair 14 to compare the GLTF skeleton's exports against.



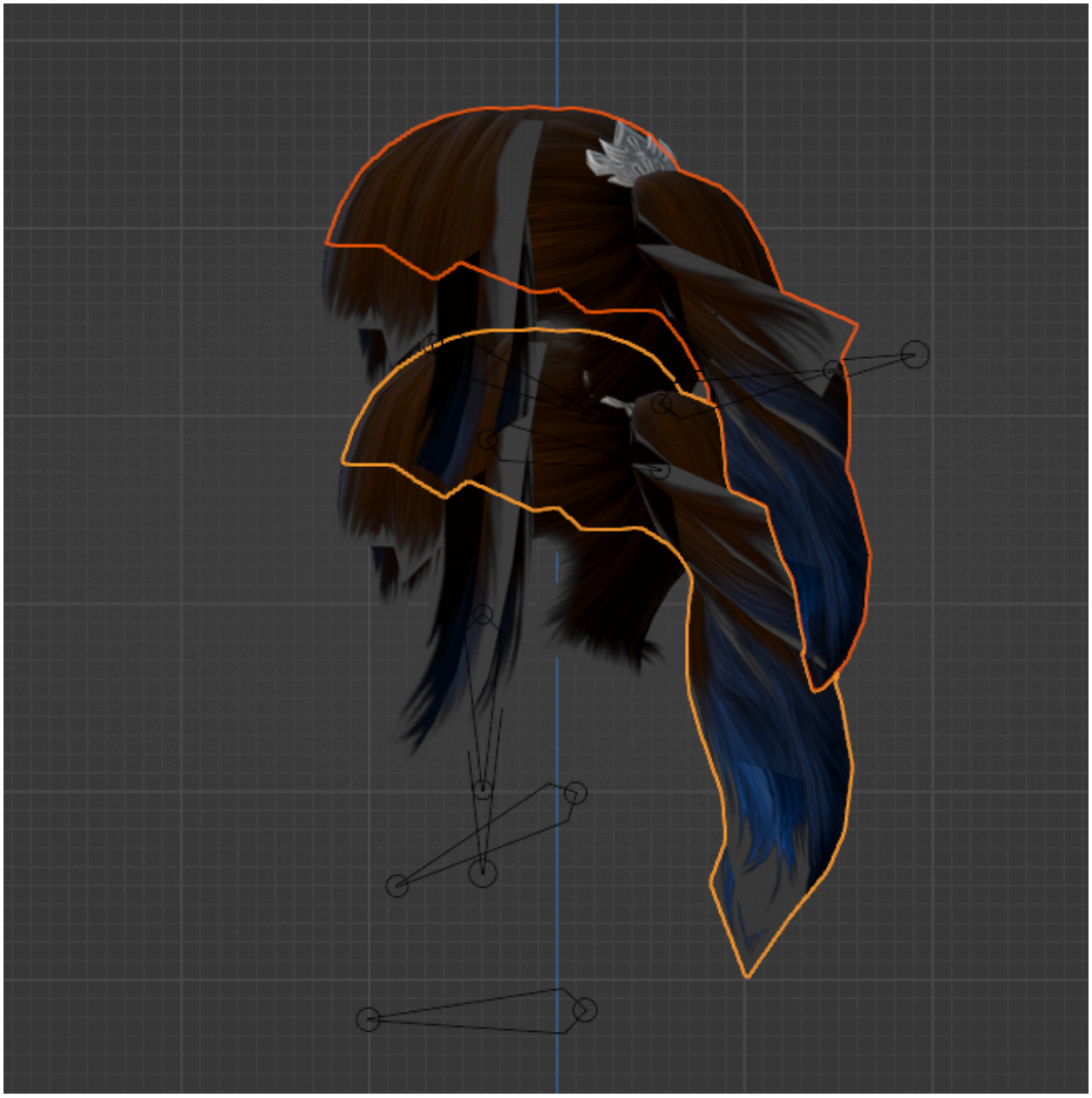
The next steps are tricky and there's probably smarter ways to do them! However, high level what we want to accomplish are the following:

1. Refit the F hair mesh to midlander M
2. Move the extra skeleton bones to new positions around the M hair
3. Join the extra skeleton bones with a midlander M head bone
4. Export the modified skeleton as a GLTF for VFXEdit
5. Join our extra skeleton armature with the full midlander M hair armature
6. Export the hair mesh as an FBX for Textools using our new bones and weights

phew!!!!

refit the hair

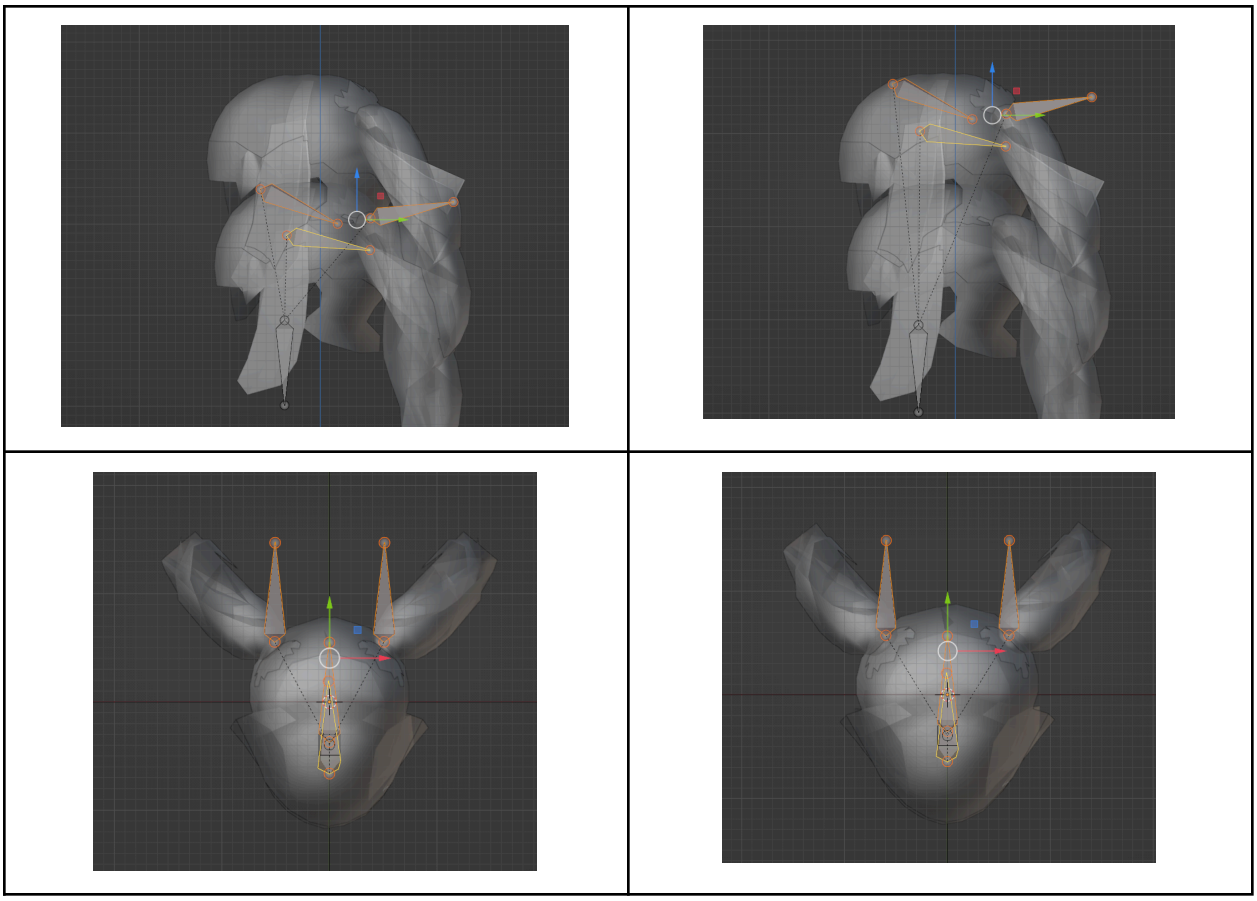
i already did this for demonstration purposes but this is as standard as any hair port or edit; you can also use an existing port instead of making a whole new one



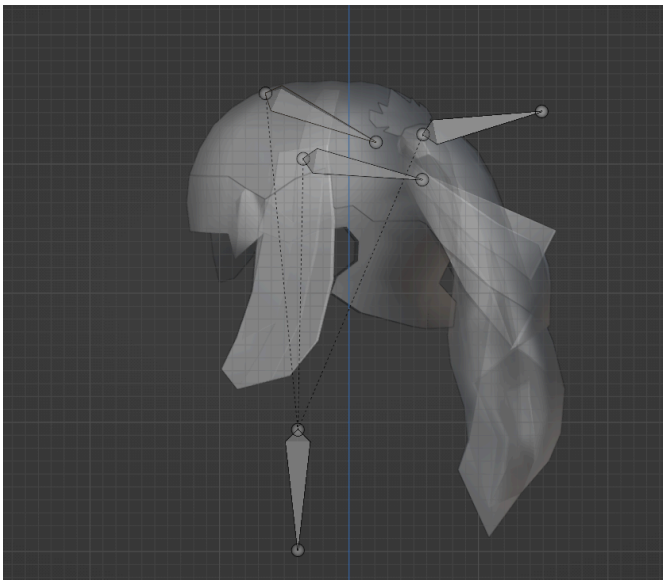
The important part is that we end up with a visual representation of where our extra skeleton bones should end up.

translate extra bones

There’s probably a smart way to do this or script this with Python, but essentially all we need to do for this step is to move our extra bones into complementary locations on the refit mesh.

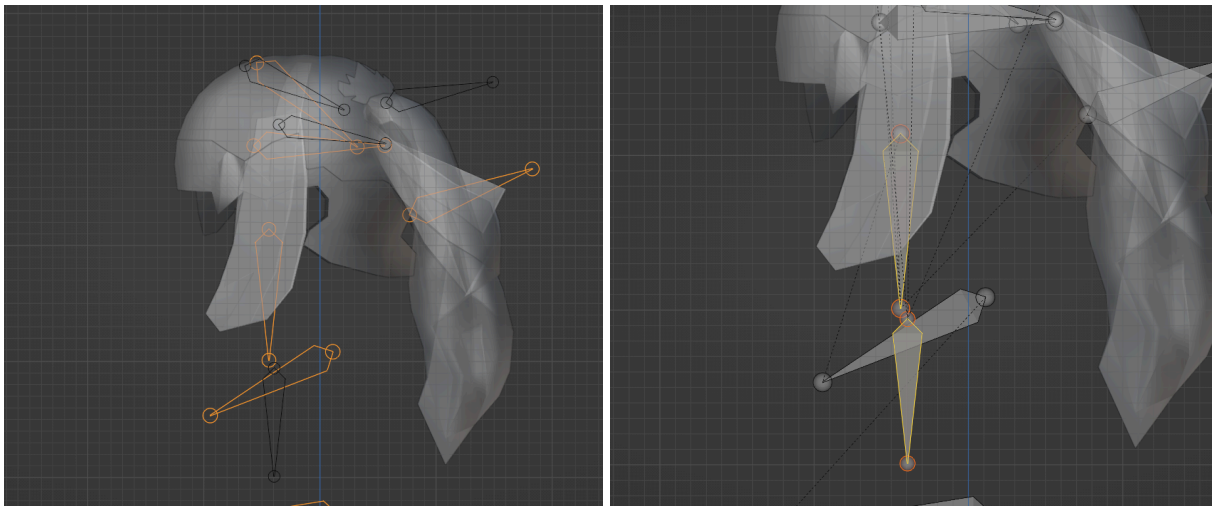


The positions don’t need to be exactsies, and they can always be adjusted later if you’re finding they’re in an inconvenient spot for posing, weighting, or physics. Don’t worry about the j_kao bone for now, as we’ll be using our target race’s head bone in the next step, instead of the one packed with our input skeleton. We should end up with something like this to begin reparenting:

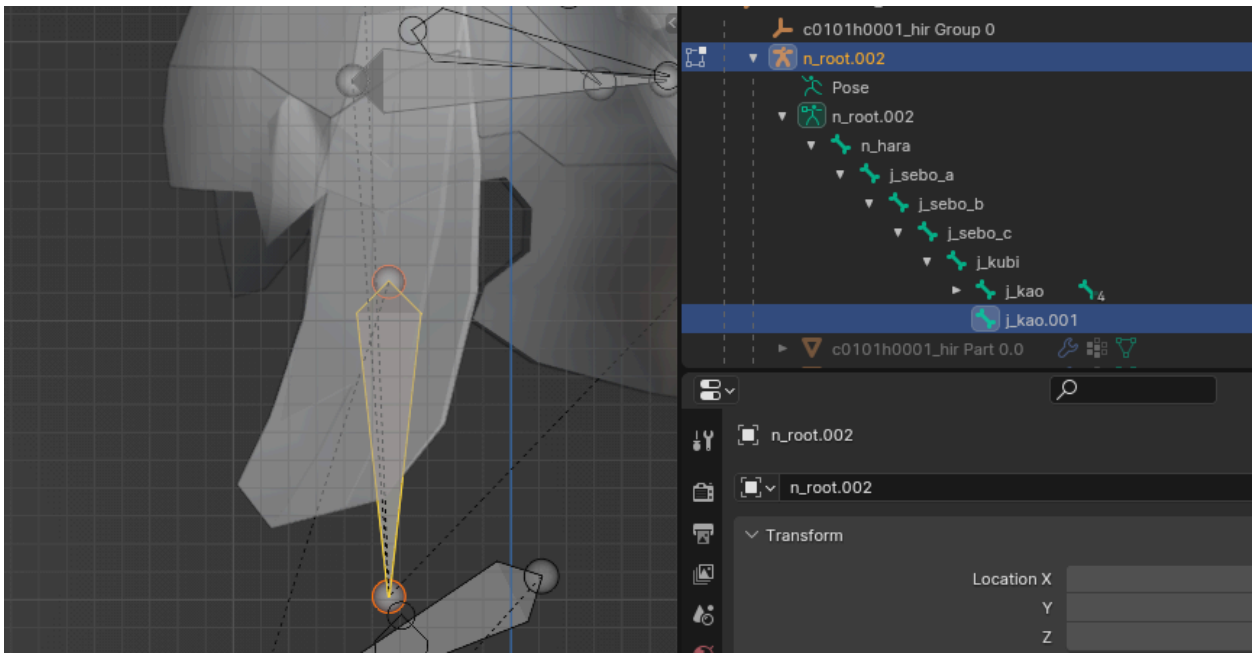


join with target race head bone

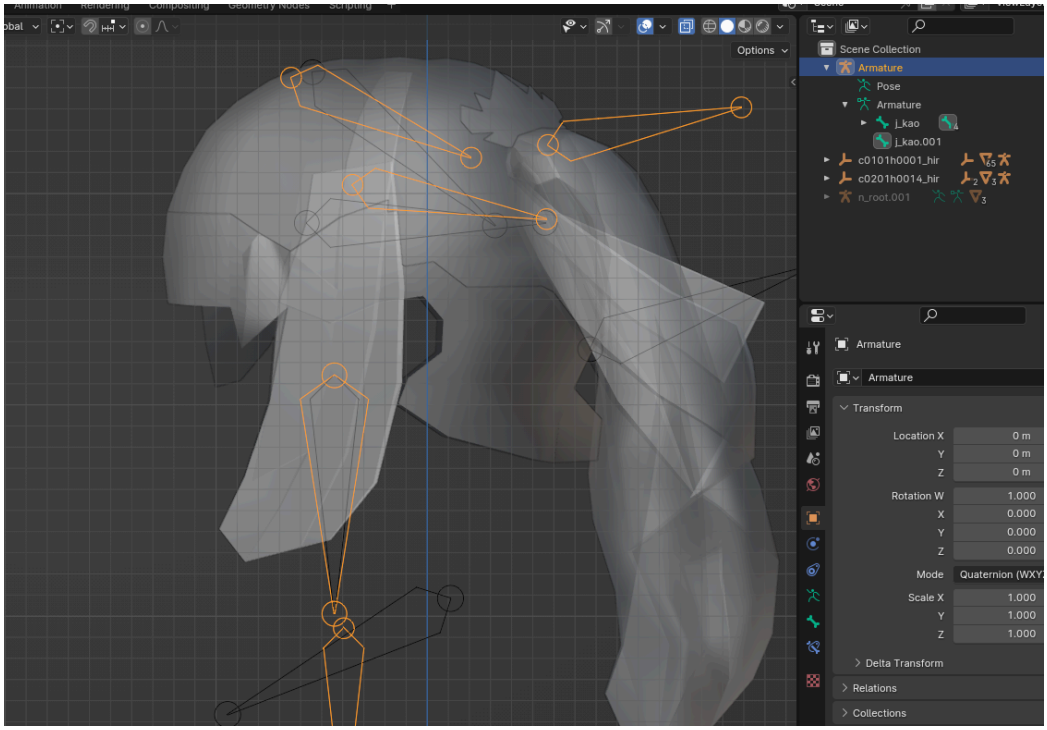
If you haven't already, import a hairstyle from your target race (midlander M in our case) to yoink the armature of. This could be your hairstyle you plan to replace, or any of them. We can hide the mesh, since all we care about are the bones:



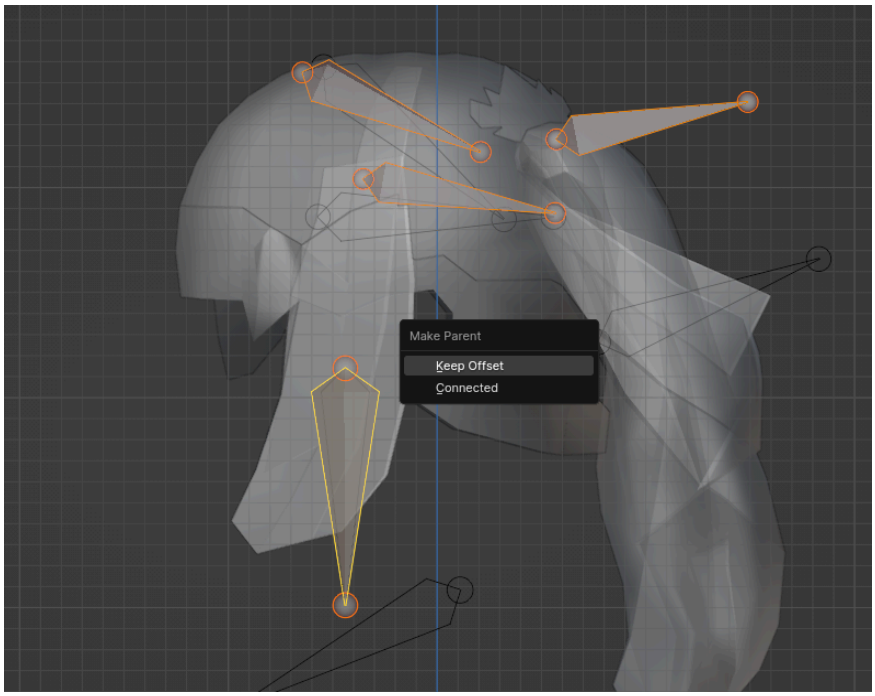
Our midlander F head bone is way below (and smaller than) the midlander M j_kao, so for consistency, we'll make sure to tie these extra bones to the masc-sized bone instead.
Select the new M armature and enter edit mode - duplicate the j_kao bone here using Shift+D.



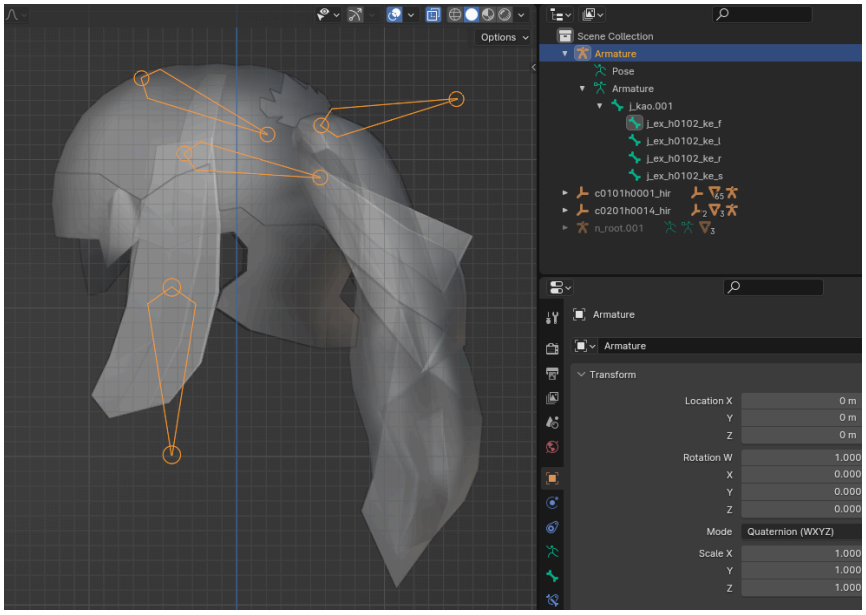
With your duped j_kao selected, press P to separate it into its own armature. Then, in object mode, join that split armature with your extra skeleton armature using Ctrl+J.



This leaves us with an extra skeleton armature containing two j_kao bones - we can delete the original j_kao in edit mode, which will unparent our ex bones. Still in edit mode, select each EX bone followed by the new j_kao, then use Ctrl+P to Make Parent for them. Make sure to use Keep Offset to maintain their floating positions.



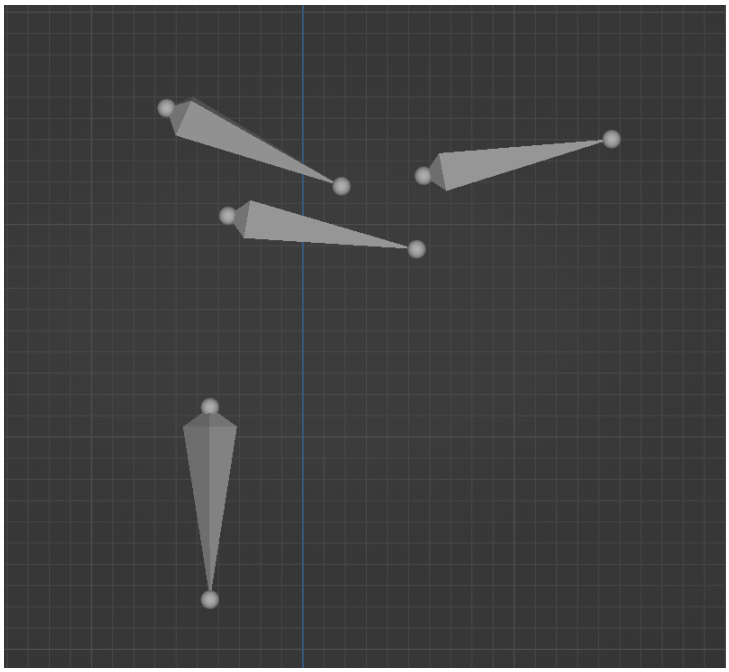
This leaves us with our repositioned head bone and extra skeleton bones, all lined up nicely with our refit hair mesh. Note that the mesh should NOT inherit from this skeleton - we export it as the armature alone for VFXEdit.



NOTE that, as a potential alternative to reparenting shenanigans, you *could* try noting down the position/scale/rotation parameters of your target head bone in VFXEdit - saving these coordinates for later, you can dump them over the unmoved j_kao bone, but this may also shift around your child bones. Play around with it!

export extra skeleton to GLTF

Use these export settings to spit out your armature alone into a GLTF file - VFXEdit will convert this into a new **sklb** to define the bone placements around your character model. You can re-import that GLTF in blender to make sure it looks alright.



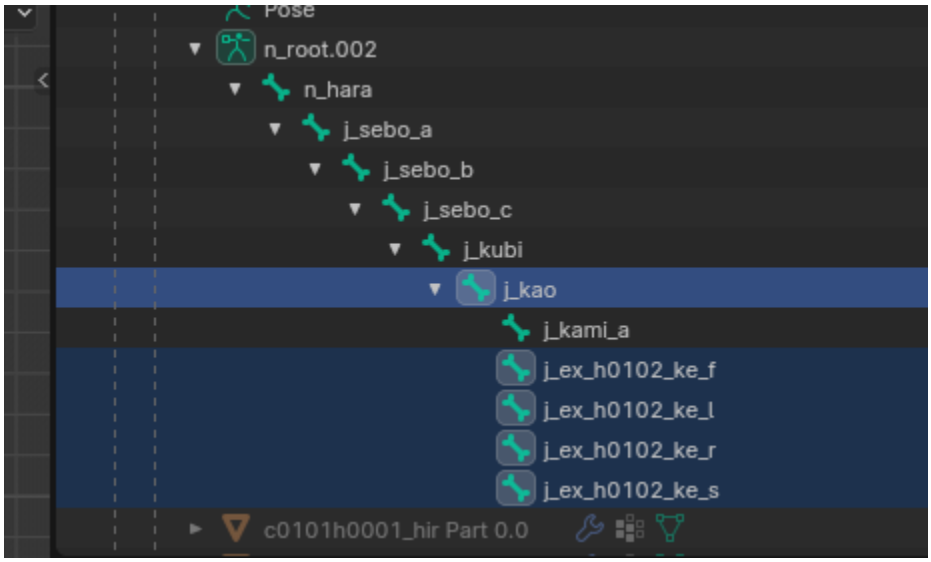
NOTE: Make sure to clean up your bone names before exporting!

joining the extra skeleton to target hair armature

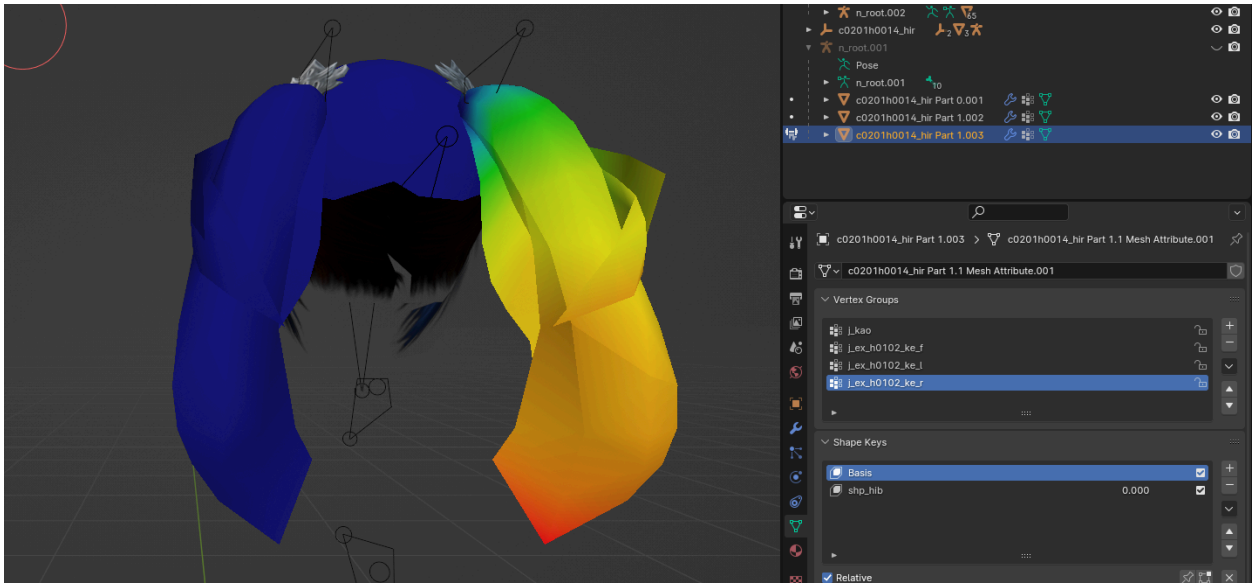
in our workspace, we should still have the midlander M hair skeleton - we'll modify this to serve as the new armature for our refit's mesh.
NOTE, you may want to save a new workspace or create a duplicate copy of your edited SKLB armature, since we'll be attaching it to the rest of the body now.

On your target hairstyle's armature (midlander M 0001 for this example), you can remove any existing ex bones in edit mode, then join our extra skeleton armature through object mode as we did before.

To resolve the two j_kao bones now, you can either delete the extra skeleton's j_kao or the M skeleton's j_kao, they should be equivalent. Re-parent your ex bones either way, so that we end up with a hierarchy like this:



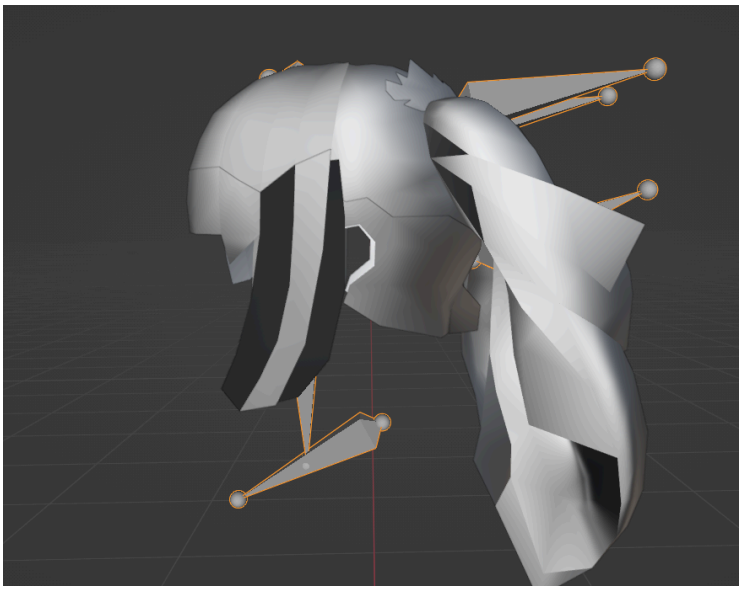
Now that our target armature is set up correctly, we can change the armature deforms on our refit mesh to use this midlander M skeleton, instead of the midlander F skeleton. Good naming/renaming conventions will help a lot here to keep straight which is which.



Our refit mesh should now be pointed to our combined armature of the hair extra skeleton and the midlander M spine+head; meaning that when we load this hairstyle with our extra skeleton configured ingame, the weights above will apply roughly the same as they do for midlander F, just on the resized midlander M model. You can now adjust your weights as you like to using our newly created extra bones.

exporting the refitted hair as FBX

with the new armature and weights (edited or original) applied, output your mesh and armature as an FBX for textools as you would with any other mod. Our final FBX should have the EX bones and weights attached.



OK so now what?

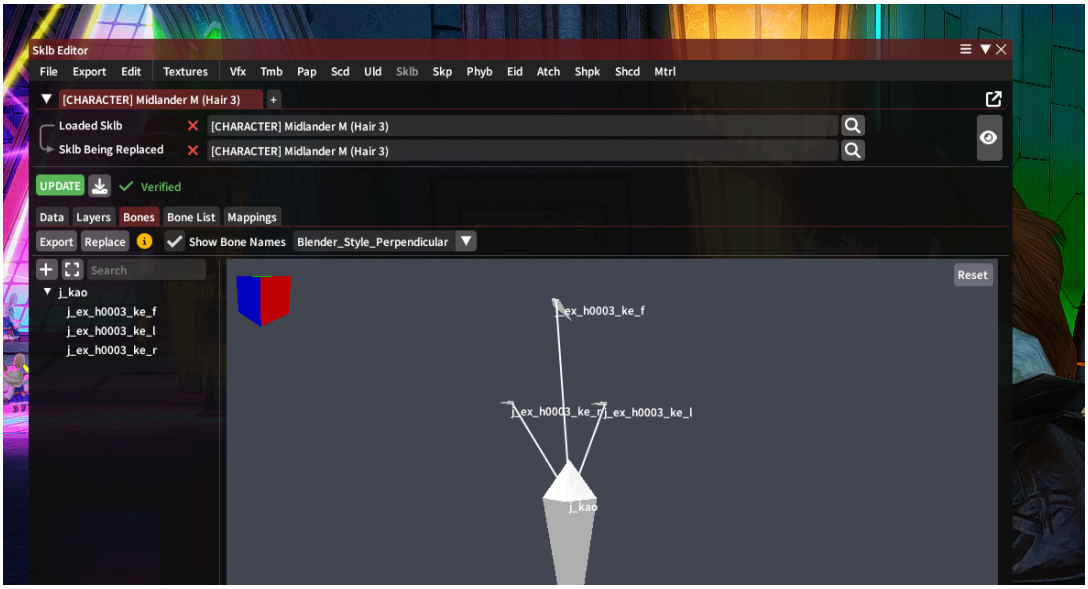
The above steps have produced the following:

- a .GLTF file containing the edited extra skeleton
- an .FBX file containing the edited mesh, weighted to the extra skeleton

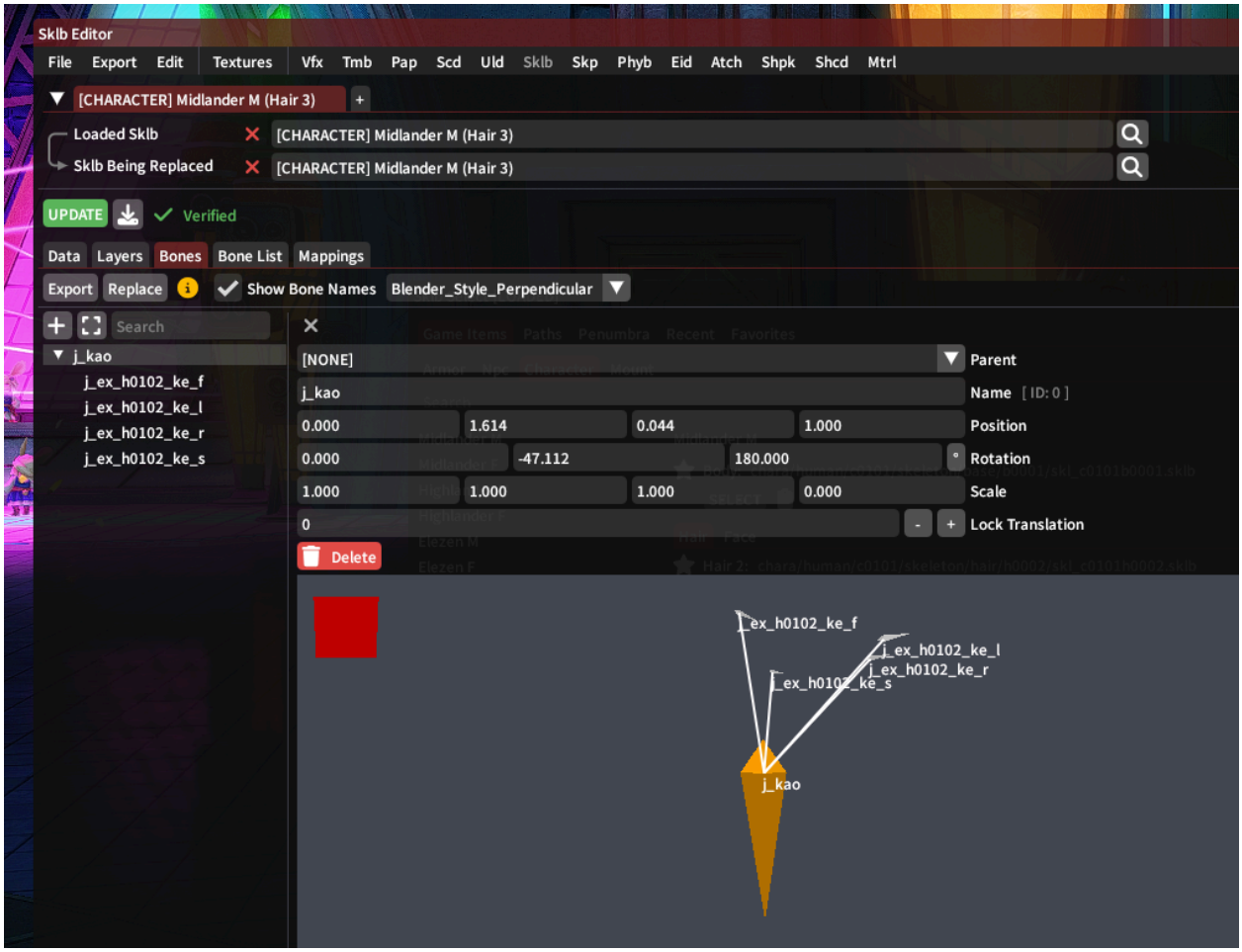
We can technically load the FBX as a new mod at any point now, but because we haven't yet configured our edited extra skeleton, it won't be able to load our extra bones or the weights we've applied to them - ingame, we'll only see whatever extra bones our target hairstyle originally had.

Not ideal! To take advantage of our new skeleton and bone weights, we need to go back to VFXEdit. This time, we should load the extra skeleton of our target hairstyle (midlander M 0003 for this example.)

sklb replacement

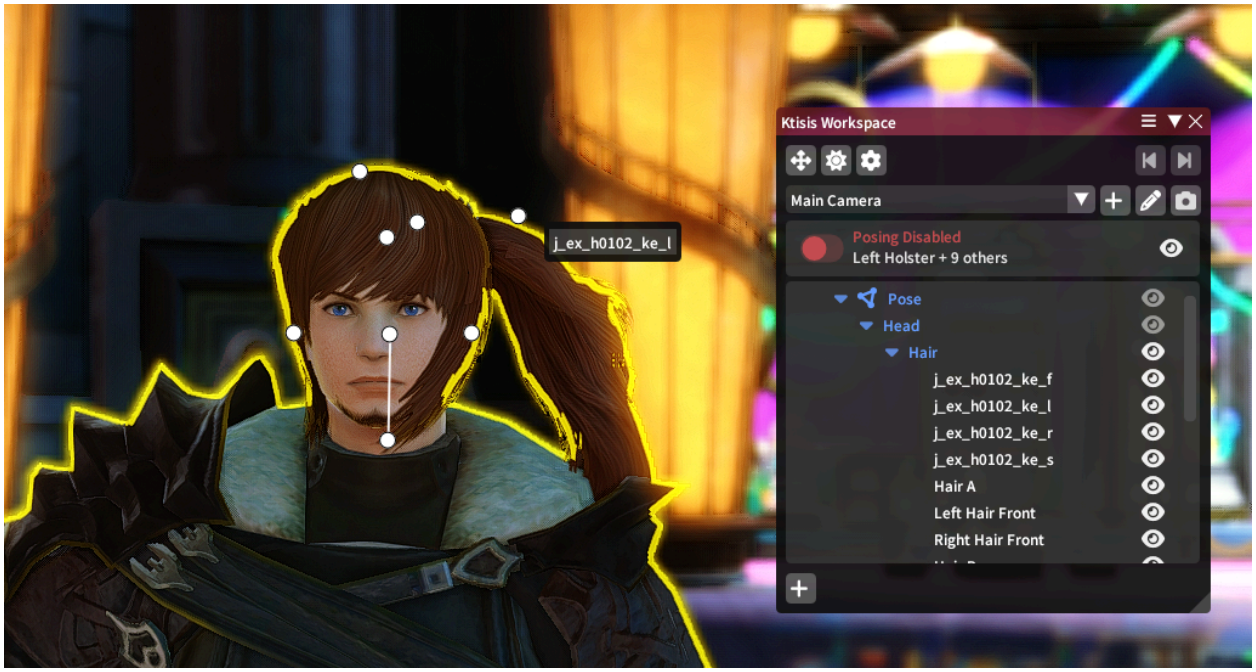


Select the same hairstyle for both Loaded Sklb and Sklb Being Replaced - this means that when we import a new skeleton over Midlander M 3, we'll be able to see the preview live. Hit **Replace** and pick out your edited GLTF armature.



Important Aside: each bone in an SKLB has both a unique Name and a unique ID - the latter *cannot* easily be changed, but is essential for how the game interprets parent-child relationships. If bone B is a child of bone A, make sure that bone B has a higher ID value than bone A. You can read a technique for ensuring sane bone IDs in the addenda here.

Once the skeleton is loaded, hitting the **UPDATE** button in VFXEdit will apply it to our playermodel, and the new bones can be viewed in gpose (provided your character has the correct hairstyle and metadata for loading it)



Oopsie boingo!!!!!! In my case, something got screwed up with the rotations and positions of the bones before I exported them from blender, meaning that everything loaded on a roughly 45 degree tilt. But! We've got bones! And they have weights!

A bit of trial and error with editing and reimporting later, and we have the bones loaded in the correct positions (my j_kao was rotated to -47 roll degrees in Blender, needed to be 90).

You'll likely notice that weights are a bit janky at this step, and that's just fine! Now that we've set up the sklb with bone placements that we're happy with, we can revisit weights after packing together the VFX files. For now, let's also set up the **.phyb** replacement - this will port the physics from the F midlander model to these bones on the M model.

phyb replacement

In the same vfxeditor workspace (you can save a workspace file locally to come back to your changes at any time), open up the Phyb editor. For the **Loaded Phyb**, select the phyb corresponding to your original extra skeleton (for this example, F midlander 102). For the **Phyb Being Replaced**, select the phyb of your target skeleton (for this example, M midlander 003).

Much like when we replace files in textools or penumbra, this configuration sets us up to overwrite the original M Midlander 03 physics with the physics of F Midlander 102. There's a lot going on in this editor that I don't even understand, but here's a bit of info:

- the Collision tab defines various 3d shapes that collision simulation will be applied against from the game's havok physics engine
- the Simulation tab can define multiple Simulators. For hair skeletons, each one typically defines a Chain of bones, where each bone is a Node on that chain.
    - Chains can be part of collision simulations, or otherwise be used to define the effects of motion on linked sets of bones. A long ponytail may have a Chain of 3 bones that run down its length, with the parameters inside each Node controlling

how they're allowed to move or be affected by motion in relation to the other bones in the Chain.

There's a lot more documentation on phybs here, but tweaking these is firmly Fuck Around And Find Out Territory. When we're porting a hairstyle and skeleton together though, the chains and collision simulations should be close enough between our origin and target skeletons that we don't need to futz with them much. We can fully reuse the twintails' physics simulation for this port, since we haven't added to the mesh or greatly changed the position of any bones.

After defining the loaded and replaced phyb, hitting **UPDATE** will let us see the physics in motion, working together with the skeleton we've already replaced in the workspace.

.phyb OFF

.phyb ON

*(**Note:** remember that the game's physics simulation is framerate dependent - you'll see less motion and effect of physics on these bones at higher framerates. The above were taken with FPS capped to 45)*

sklb and phyb export

With these changes applied, we can export them both at once into a .PMP, .TTMP2, or both. Once you're done making changes, save your workspace if you want to come back to it, then hit Export. Choose your output format, and in the export panel, make sure to check off Sklb and Phyb.

From here, just pack up your sklb+phyb mod together with your weighted mesh, and you're done! waow

example files

heres's some uploads of various files from this example project!

- Refit hair FBX with F armature
- Refit hair FBX with expanded M armature
- Original GLTF skeleton
- Edited GLTF skeleton
- Final physics+sklb PMP, TTMP2

FAQ

How do I unfuck my bone IDs after replacing the skeleton?

Great question!!! I don't know!!! SKLBs are a nonsense format dreamed up by unhinged developers with the havok engine!!! However there's a method to preemptively unfuck your bone IDs ***before*** importing the edited skeleton. VFXEdit creates new bones in incrementing

order by ID, and when you import a new skeleton, it attempts to match any bones on the imported skeleton to bones that already exist on the targeted skeleton. This means that we can rename/create as many bones as we like before import, line up the parent/child hierarchies, *then* bring in the new mesh.

Bones can be added into the hierarchy arbitrarily before import to make sure your IDs are in the correct order, then renamed to map your imported bones onto those correct ID values.

What if I want to use bones from multiple extra skeletons?

Sure, you can do that! Important things to keep in mind:

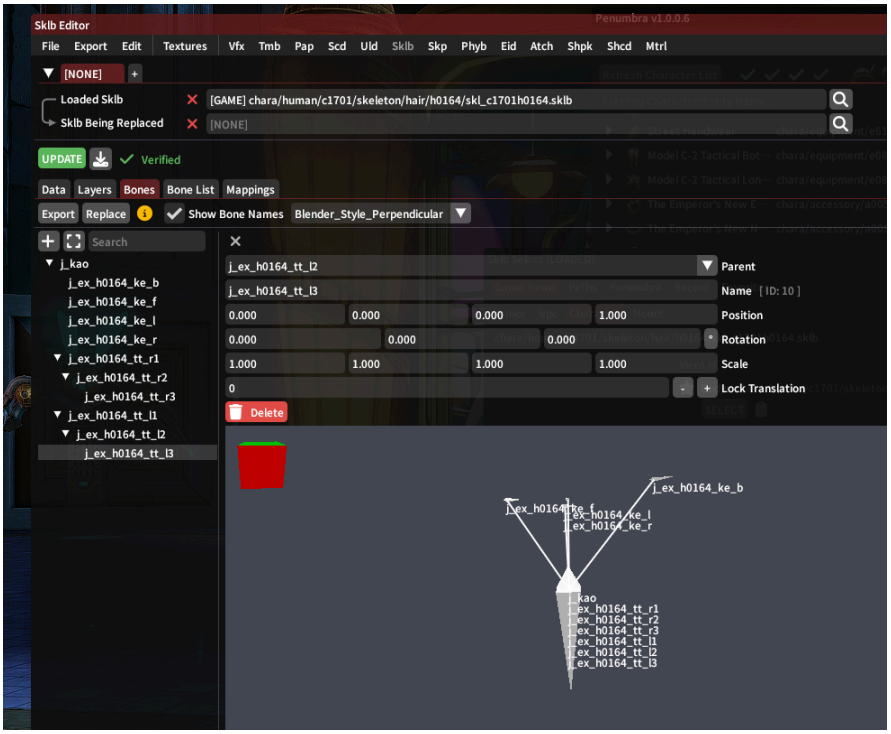
- A hair can only have one active extra skeleton at a time, so your various extra bones will need to be condensed into one sklb.
- A hair can only have one phyb applied to it at a time, so if you want all your combined extra bones to have physics simulation, you'll need to muck with a phyb and try to combine the various chains/colliders from source phybs into one that references all of your bones.
    - You can also just do one phyb or no phyb, and leave the extra bones there for posing purposes only (once weighted on the mesh)
    - Too much physics all at once (especially with collider physics) can make really weird/laggy/game breaky shit happen, so try to keep it reasonable

What if I want bones that don’t exist on ANY extra skeleton?

Yeah sure you can do that too! A caveat would be that adding physics to fully new extra skeleton bones would require making a custom .phyb to affect them (I am scared of the phyb menu so please don’t ask me), but for posing purposes you can spin up some new bones real easy.

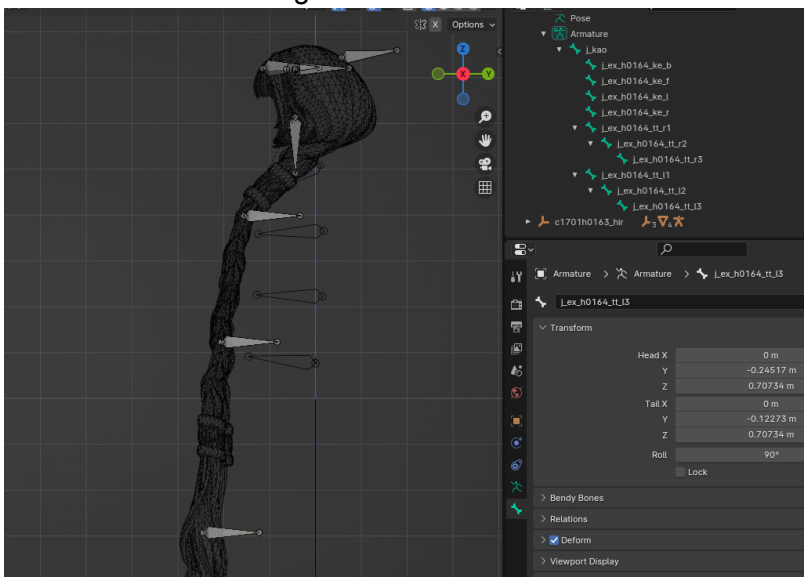
EDITOR’S NOTE: this is not actually ‘real easy’ but is definitely doable! Took about a day to get the below mini-tutorial put together, with some trial and error both in weights and skeleton settings.

Identify your in-use extra skeleton

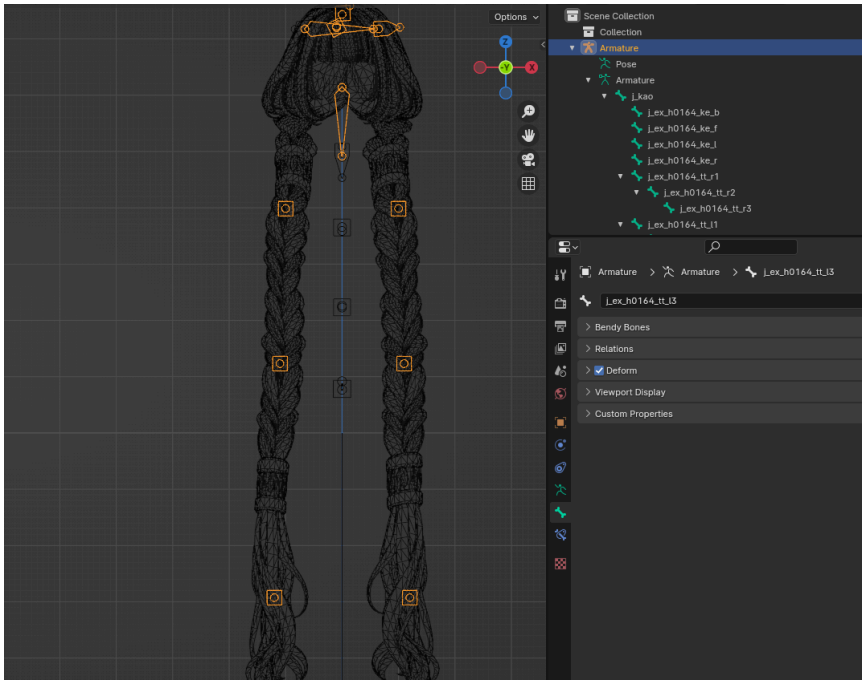


Add and organize additional bones in the hierarchy; choose a sensible naming scheme. For this example, I use *tt_rx* / *tt_lx* for twintail right and left bones. Export this GLTF when you have your bones named and parented like you want. Keep in mind the importance of bone IDs!

Import your new GLTF and target hair mod together; then, reposition your newly made bones along the hair as needed.

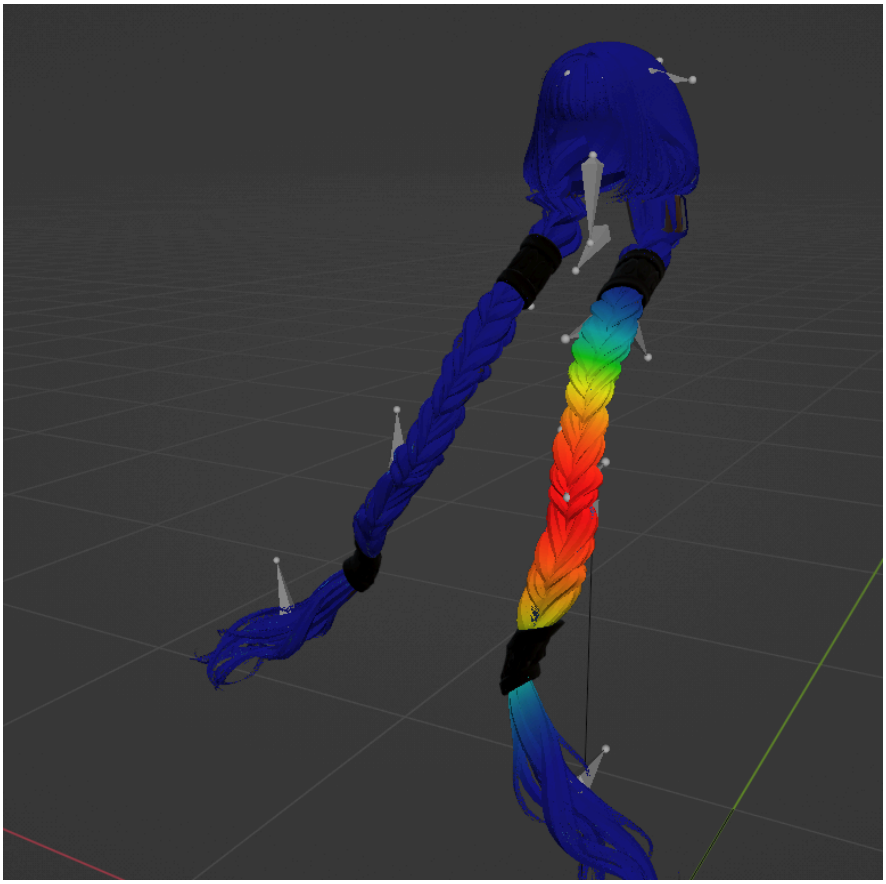


Remember to keep in mind bone direction, as this will inform how different angles make the mesh bend.



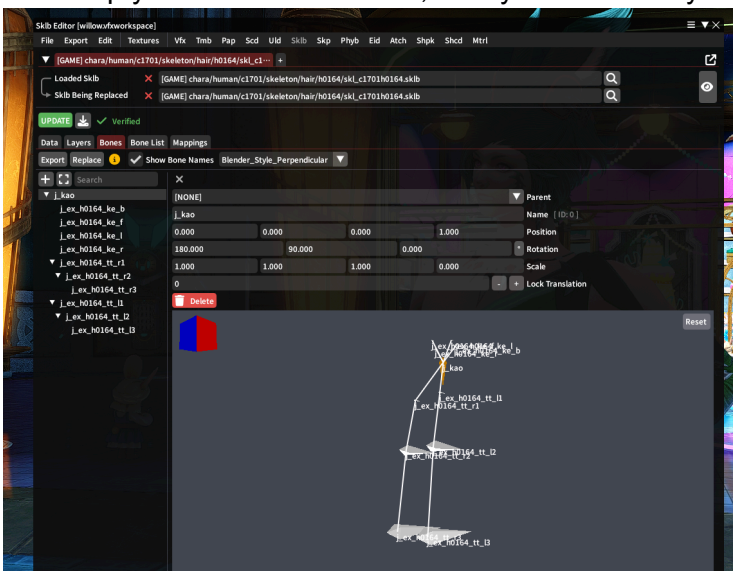
Keep in mind the X-axis distances as well as the Y/Z positioning. Move symmetric bones the same distances apart for consistency.
Once you've moved your bones to new positions, you can export this armature to GLTF, then parent it to the imported hair's skeleton as described above.

Weight paint! Can't help you out here, use your best judgment. I'm bad at it!!!!



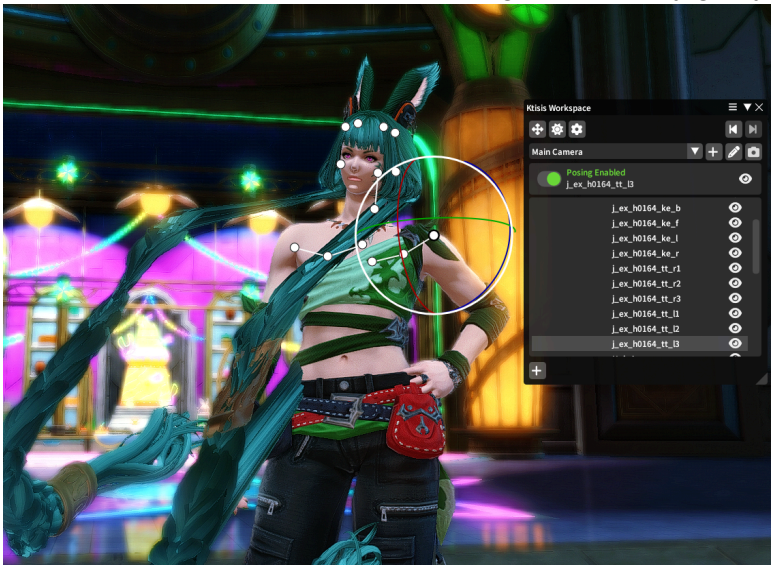
Once you've weighted satisfactorily to the new bones (you can test it out with **Pose Mode** if you want to observe the horrors), export your mesh and combined armature as FBX for textools. Don't forget to normalize all/limit total!

Load up your weighted model through textools, and ingame, apply your edited skeleton over the targeted extra skeleton. Remember to set up your bone names and hierarchy beforehand to keep your bone IDs in order, if they aren't already!



Bit of fiddling later and it looks pretty OK in the sklb view!

Without the new model installed, the weights are pretty goofy…



Hm, not good!!!! I did say I was bad at this

Take two with better normalizing and weight-fixing and normal recalculating and skeleton rebuilding… (for my case, I had to offset the j_kao bone by ~1.6 Y coordinates in vfxedit to get it looking OK). To avoid jank like the above (or hair stretching way above/below the player's head), make sure that your final skeleton's j_kao bone is in the same coordinate position as the original skeleton's. Mine was set to 0,0,0.

Wow, a wholeass proof of concept!!

NOTE that these arbitrarily-created bones will not be affected by any physics unless defined as such in the accompanying .phyb file - screw around with that at your own risk, because for

complicated skeletons like this, it'll certainly need trial and error. For a modification to or similar-to an existing skeleton, you can likely copy & tweak some settings from a similar phyb and tag them onto your new bones instead. Work smarter not harder

Do I have to use VFXEdit for this?

Not technically! You could apply a similar workflow fully in Blender/Textools by using BlenderAssist - made to support animation and skeleton editing, BlenderAssist can translate an exported SKLB from FFXIV to a GLTF for use in Blender. The size, scaling, and rotations can all get a bit funky in the process, and it isn't exactly currently supported compared to VFXEditor's development, but it can be a useful tool in your toolbox!

special thanks

wouldn't be possible to write or coherently read a guide like this without the help of

- Modding 101
- Students of Baldesion
- Saenomaed
- Ro for lending their sample mesh
- IrisStarcaller my bones mentor

more questions? Message me on twitter or in the modding 101 discord!