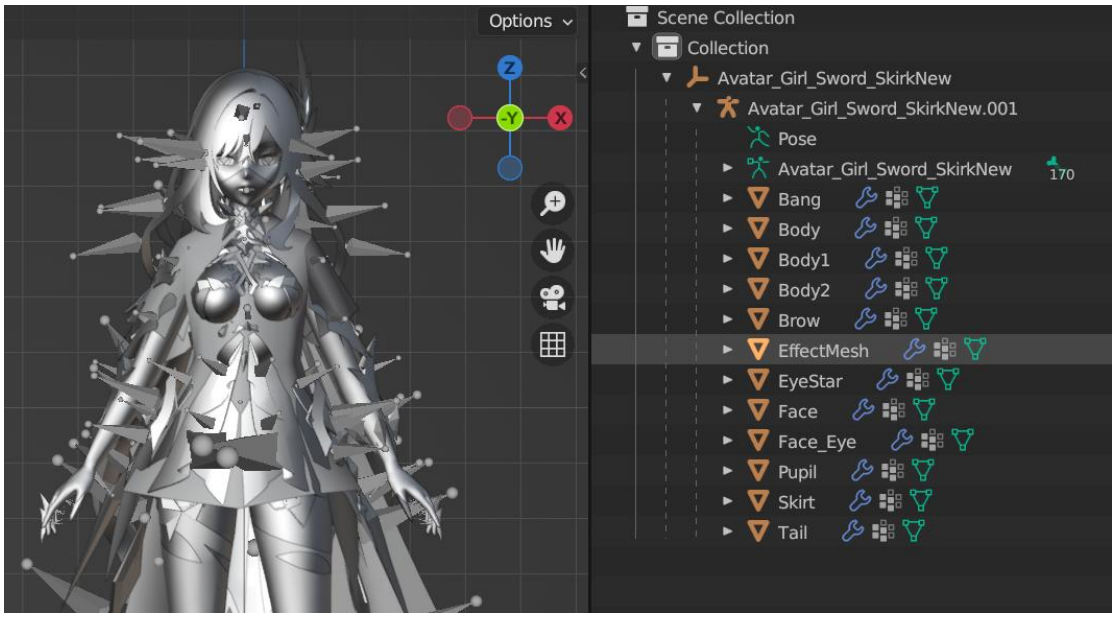
Get an armature for your mod, so that you can reuse your mod model in scenery mods, in blender scenes, check your VG weights, or whatever you want to do.

Thanks to zeroruka. I referenced the VG name mapping method from his script (here)

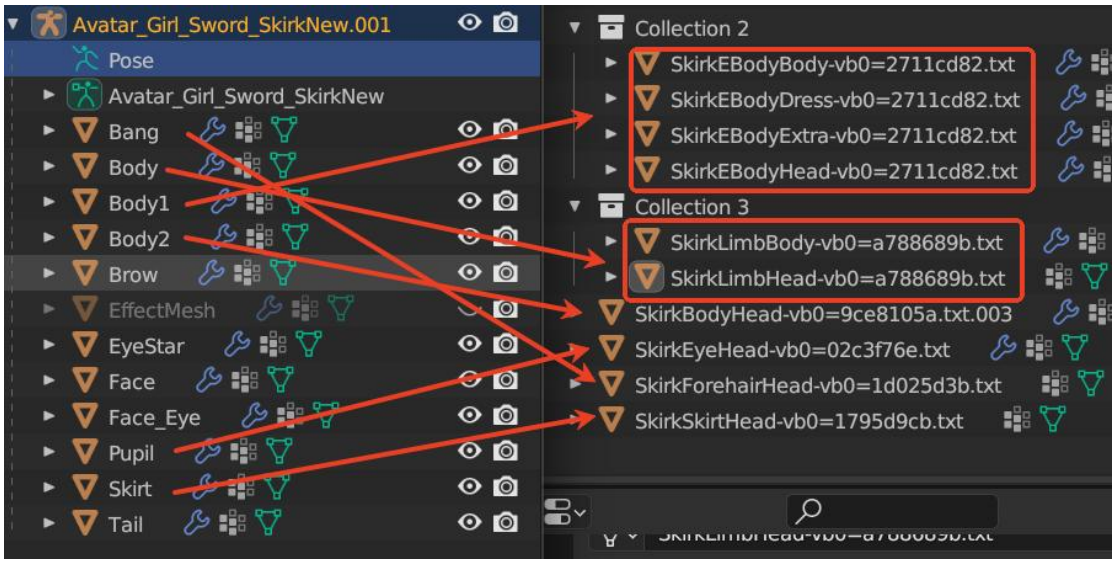
# 1. Get a character assets from game unpack

One of the place You can find them is the HoyoToon Discord server (invite link). They provide characters or NPCs' assets for GI/HSR/ZZZ/WW.

I'll use Skirk's asset an example. The asset is in FBX format. After importing it into blender, you will find an armature, some meshes, and a lot of empty axis there. Feel free to delete the axis because we are not going to use them.

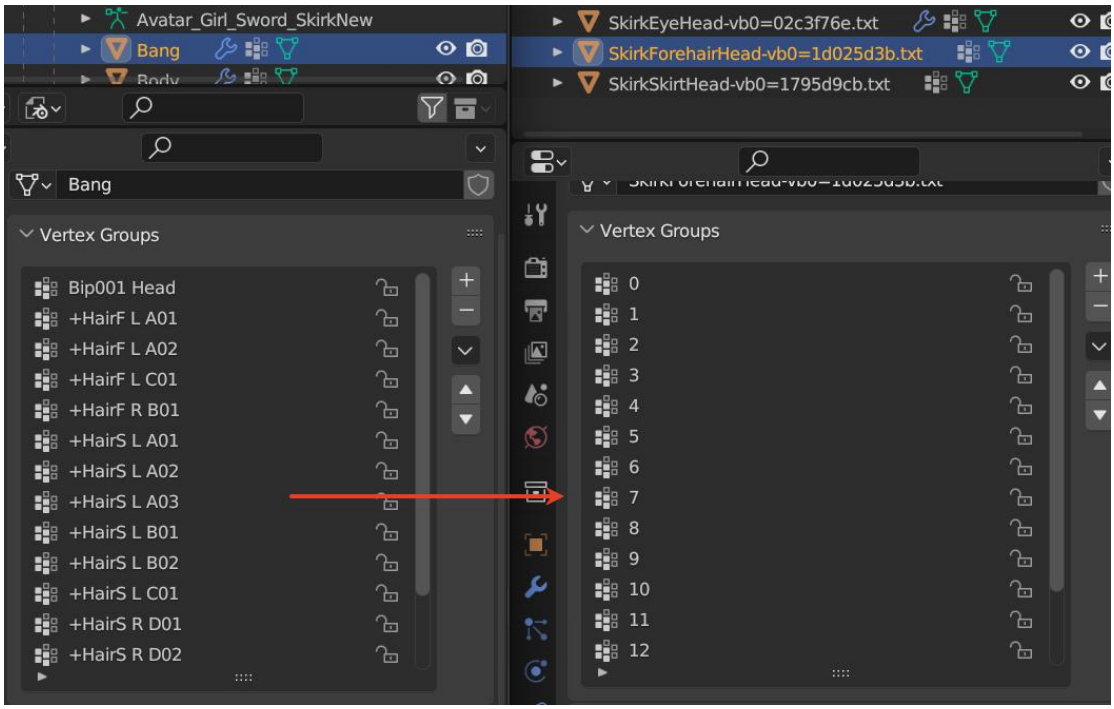



Each one of these asset meshes correspond to one IB hash in mod. (LoD is out of consider.) Some of the asset meshes have more than one material. Each material corresponds to one mesh in mod.




Mod meshes dumped by 3dmigoto is a data structure compressed by unity and passing to

d3d11. The VG names of the asset meshes are arranged and mapped to integers during the compress process. Therefore we are seeing a integer format VG name in the mod meshes. What we are going to do is to rename the bones according to this mapping, so that these bones can control the mod meshes.




## 2. (optional) Rotate the bones

I don't know why these bones in the asset armature are pointing outside, instead of pointing their child, and i'm uncomfortable with it.

The script "bones_rotate_90_z.py", as it's name suggests, will rotate the edit bones and make it pointing to it's child like MMD armatures. Select the armature and run the script. The result looks like this.

# 3. Add control bones for mod model

The script “rename_bones.py” will add control bones for mod model. But before running the script, you need to set the 3 input list manually: mesh_list, prefix_list, layer_list.

“mesh_list” contains the name of meshes you need in the mod model. There are some mesh that usually won’t be included in a mod, like Face, Face_Eye, Brow (These form the face of the character) and EffectMesh (i don’t know what this mesh is for), and they don’t need to be in the mesh_list either.

“prefix_list” will add a prefix to the control bones. It is prepared for characters whose body is consist of multiple meshes. Not every bones in the armature is included in the mesh’s vertex groups, and this will lead to a different name mapping. A same bone can be mapped to different integer in different mesh, and the same integer in different meshed not always point to the same bone. My solution here is to add a prefix for the control bones to tell which mesh do they control. The content of prefix has no limit, just make sure they are different with each other.

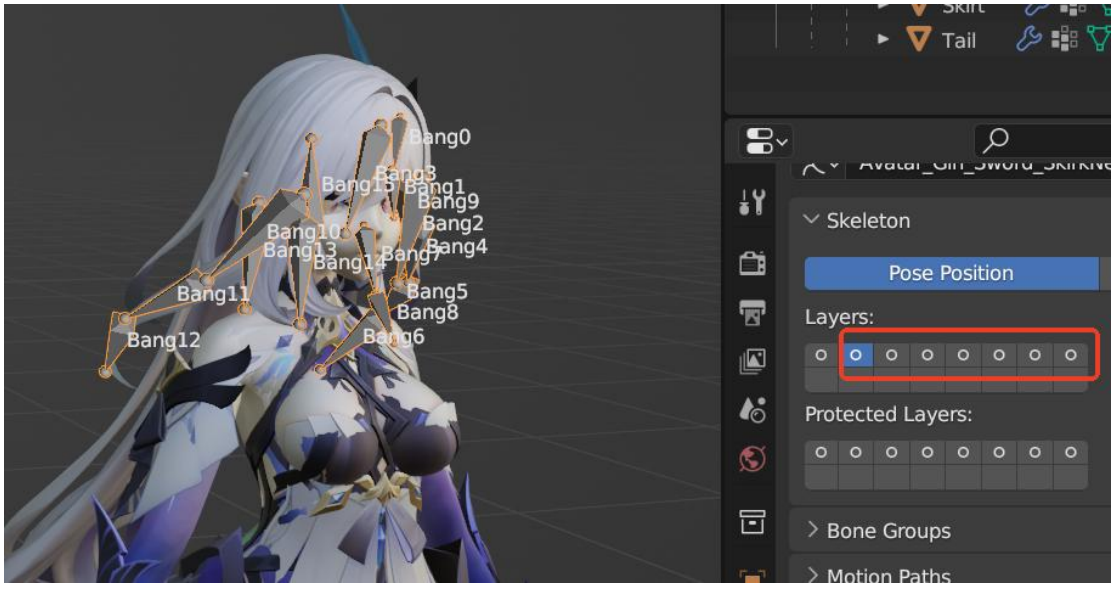
“layer_list” determinate which bone layer the the control bones are located. I recommend to put the control bones for each meshes in an individual layer.

The length and order of these 3 lists need to be the same.

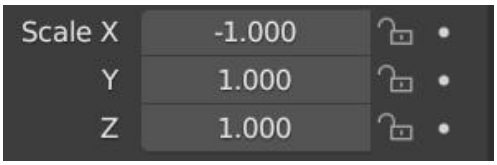
```
mesh_list = ['Bang', 'Body', 'Body1', 'Body2', 'Pupil', 'Skirt', 'Tail']
prefix_list = ['Bang', 'Body', 'Body1', 'Body2', 'Pupil', 'Skirt', 'Tail']
layer_list = [1,2,3,4,5,6,7]
```

Now run the script.

You can check the control bone for each mesh in the layer you set in the layer_list.



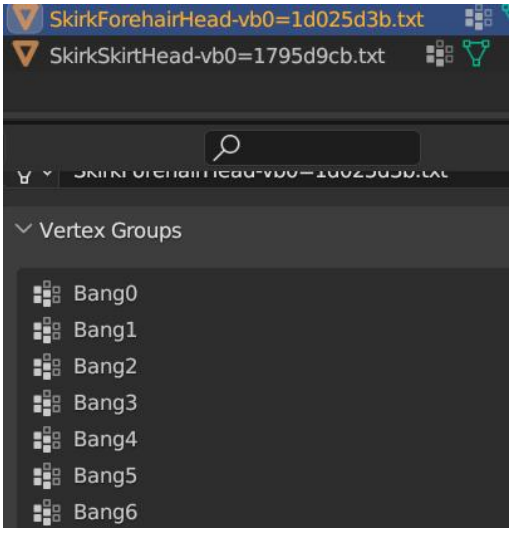

And finally, you need to mirror the armature by x axis, because the mod meshes are x-axis mirrored. But don’t apply the transform, otherwise it will somehow break the roll attribute of bones.



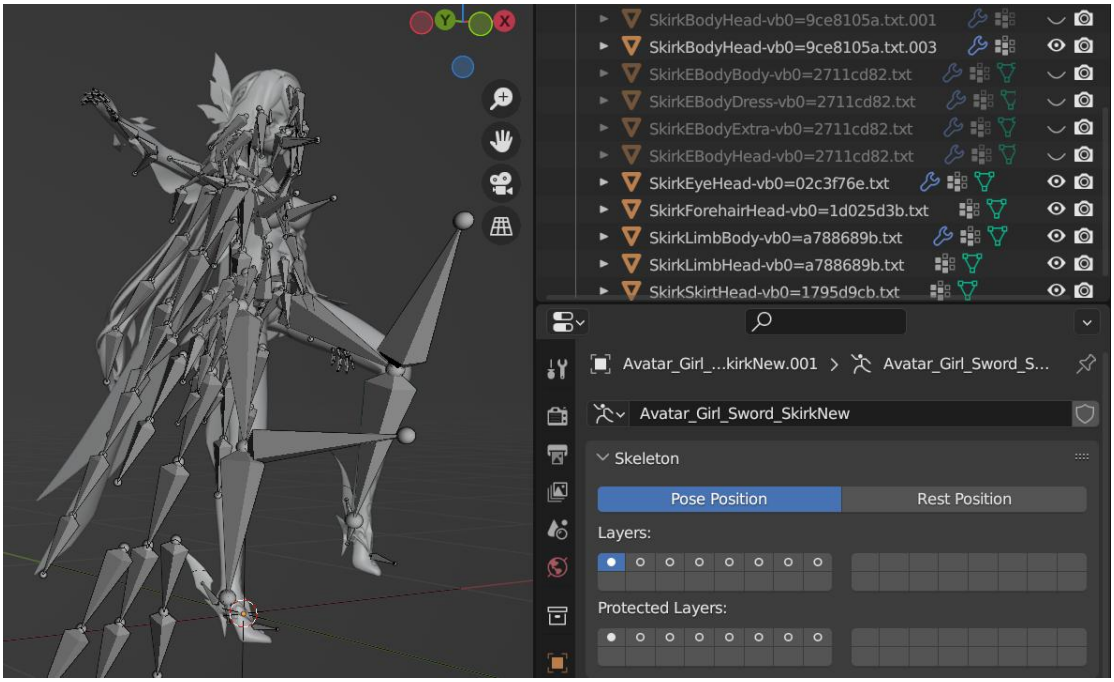
# 4. Add prefix for mod meshes

We added a prefix for the control bones in step 3. To make these bones able to control the mod mesh, we need to add the same prefix to the VG of mod meshes, too. This won't effect the mod export process, as i tested with XXMI.

The script "add_prefix_to_vg.py" can save your time. Set the "prefix" in the script, select a mesh, and run the script one by one.




Finally, add armature modifier to mod meshes and you should be able to control it now. When making pose, you should only transform the origin bones in layer 0, all control bones added in step 3 are assigned as a child of origin bones.



# Limitations

The armature from assets contains only the bones. There isn't any bone constrains like IK, making it inefficient in posing, unless you add the constrains manually. I'm still experimenting matching the asset armature with an official MMD armature. Maybe this will handle the constrains and physics all in once.