Self-Animated Accessory

Wogrim's Brief Guide to Making an Accessory with an Animation

Before Reading This Guide

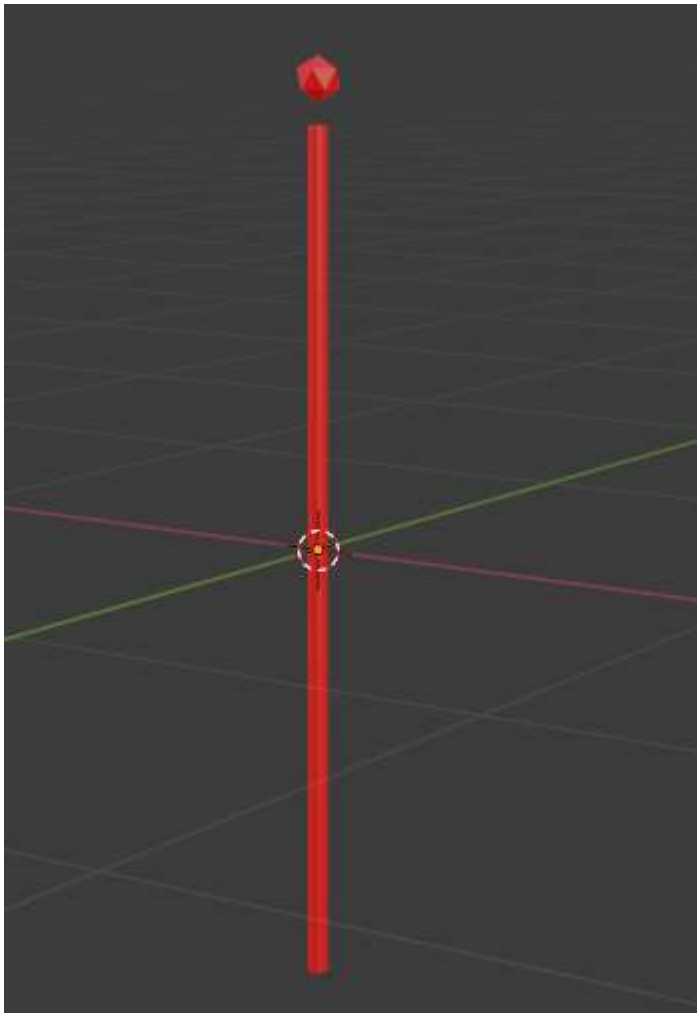
Read the Accessory guide and the Self-Animated Studio Item guide.

What Are We Making?

Similar to the self-animated Studio Item, we are making an Accessory with its own animation that constantly loops. This time around I am doing it with unskinned meshes (to show that it can be done). You will see that there is an extra complication that the Studio Item did not have.

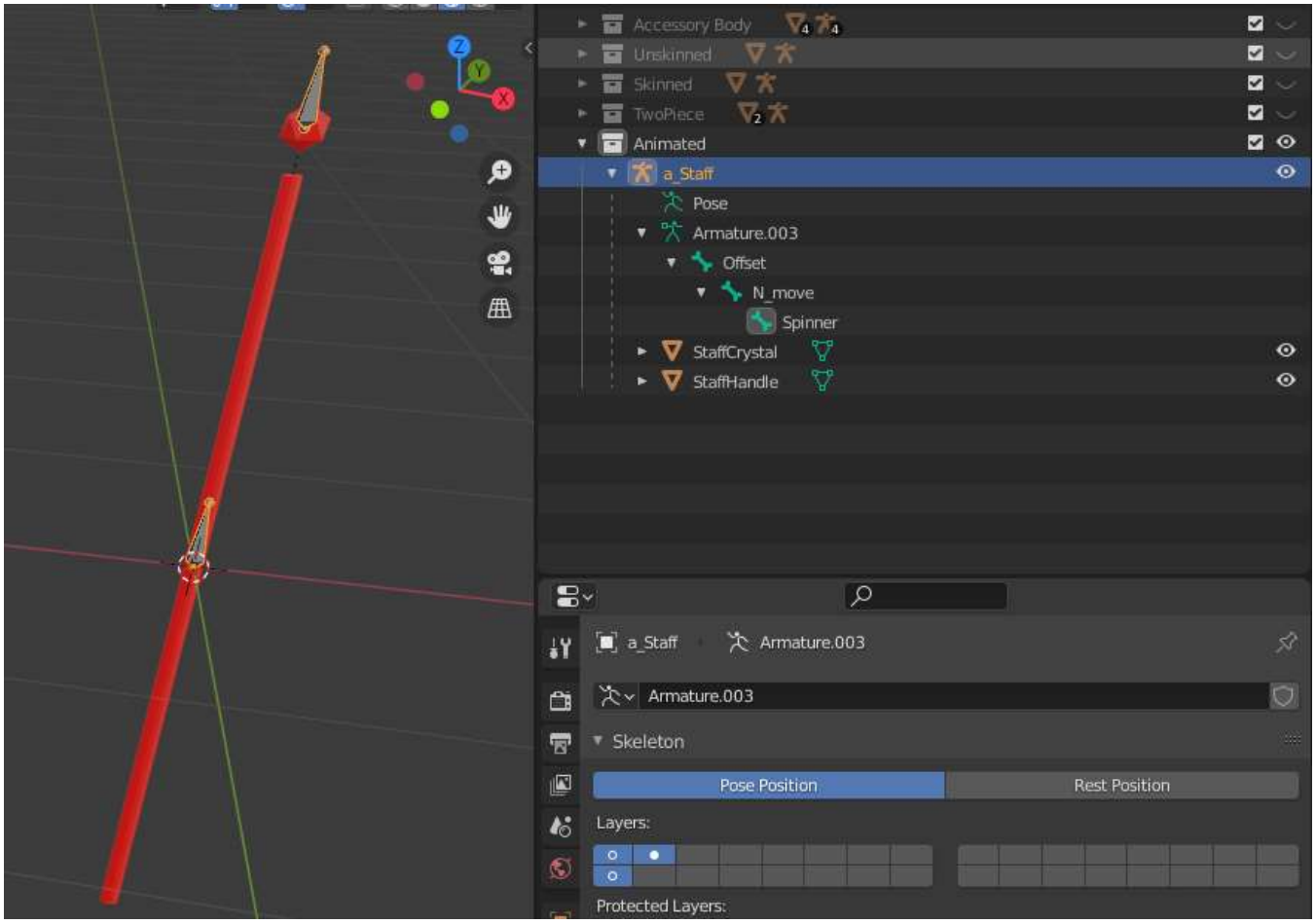
Self-Animated Accessory: Blender

A self-animated Accessory basically works the same as a self-animated Studio Item, but because Accessories are slightly more involved, there is more opportunity for something to go wrong. So here I have a simple wizard staff that will be attached to the character's back like in an RPG when not in combat.



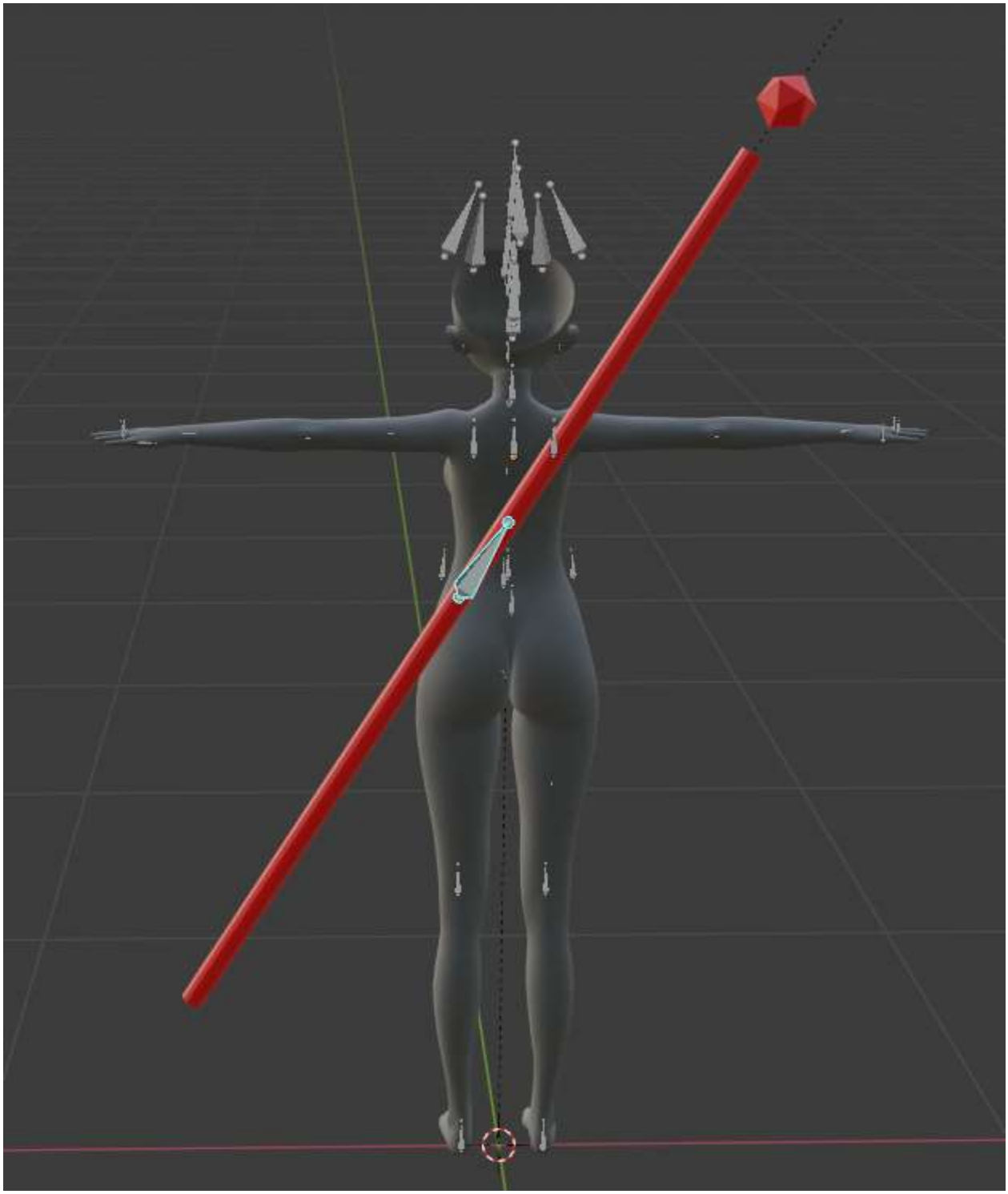
The crystal at the top is a separate mesh because it will use a different material (and shader) so it is see-through. The animation here will be that the crystal constantly spins (nothing fancy).

So I create an armature for it that looks like this:

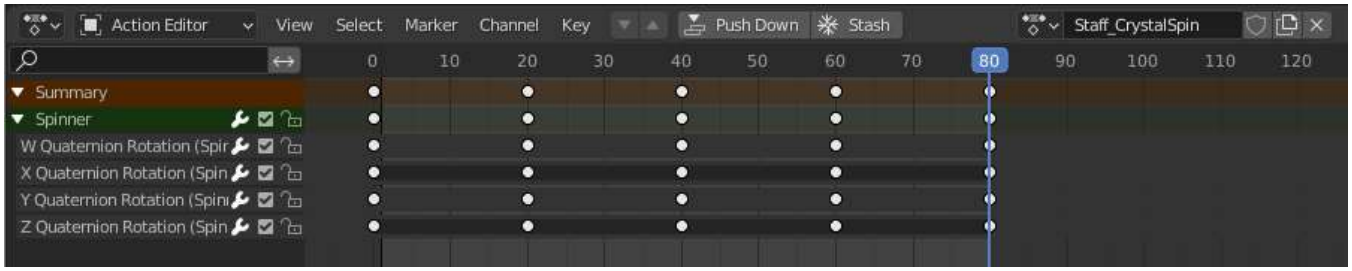


The crystal is a child of the Spinner bone, and the handle is a child of the N_move bone.

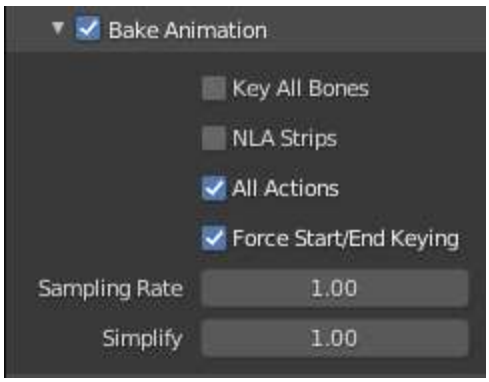
Then I show the body, put the Copy Transforms on the armature set to a_n_back, and pose the Offset bone to where I want it. Apply pose as rest pose.



So same as the self-animated Studio Item, I create an action in the Action Editor and add keyframes for the animation, but this time is super simple because I'm only rotating the Spinner bone in pose mode and adding keyframes.

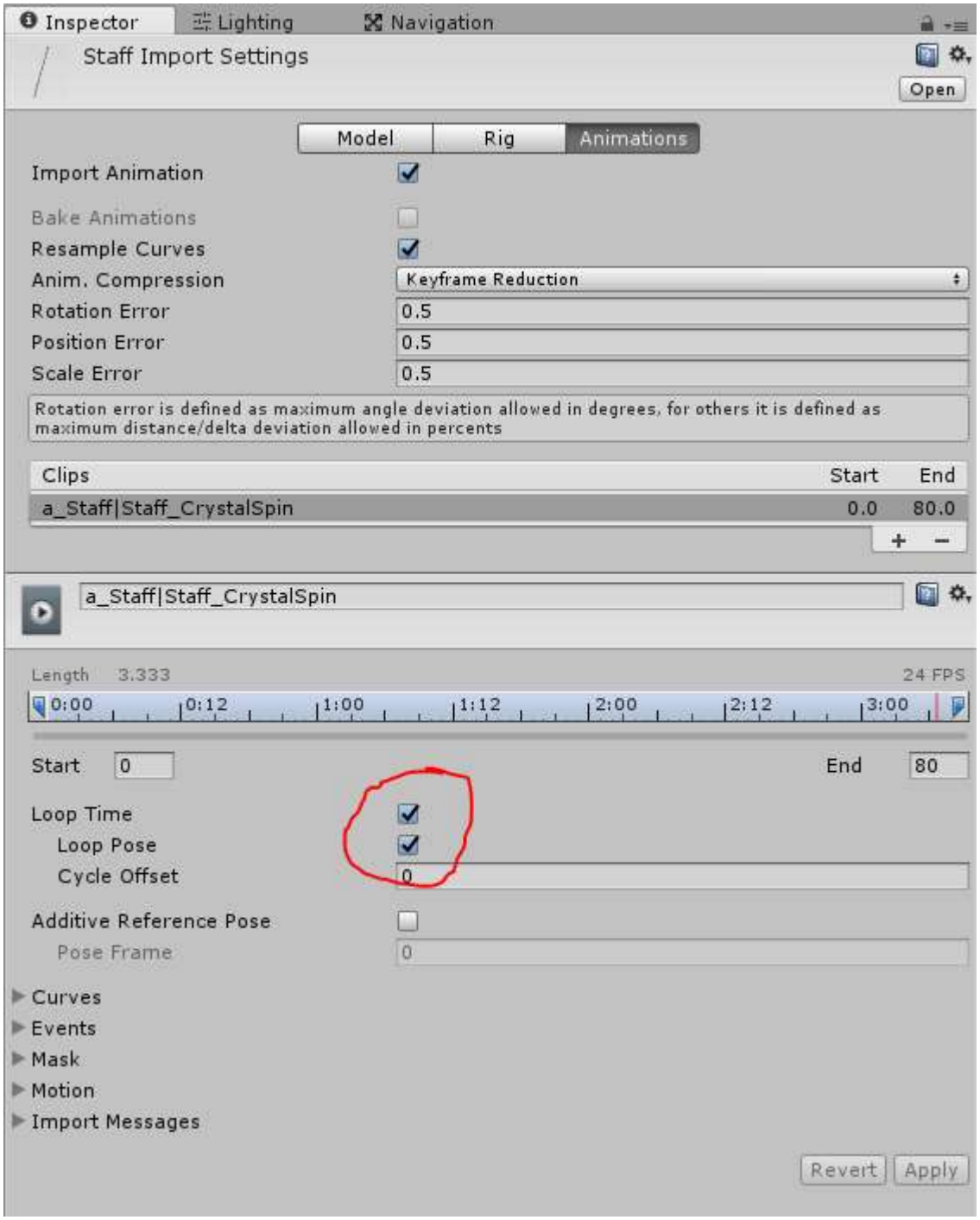


When done, disable the Copy Transforms on the armature and export your FBX. Check the animation export settings.



Self-Animated Accessory: Unity

Just like with the self-animated Studio Item, in Unity import settings I check the "Loop Time" and "Loop Pose" boxes so that the animation will loop forever instead of just playing once. Don't forget to Apply.



Do all the usual Accessory stuff.

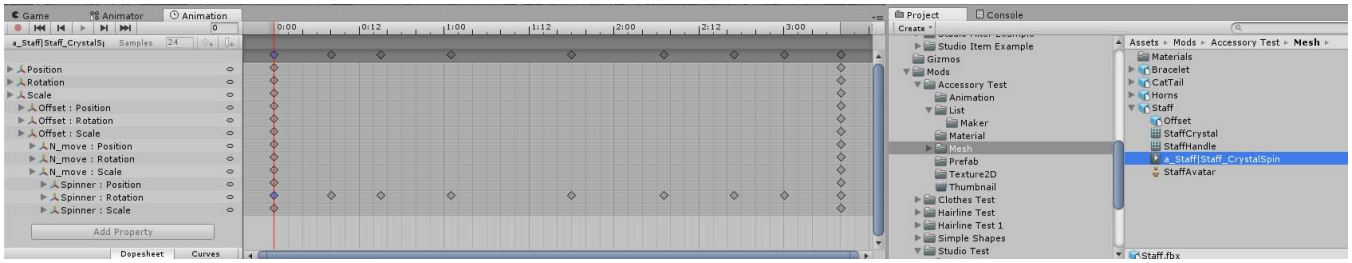
- import textures, set import settings
- create materials, set shaders, fill them out
- drag FBX into scene, zero out the rotation
- put the Accessory MB on the item, fill it out

Plus the same animation-specific stuff you'd do for creating an animated Studio Item

- create Animator Controller
- create a state
- put in your item's Animation Clip
- put the Animator Controller on your item's Animator

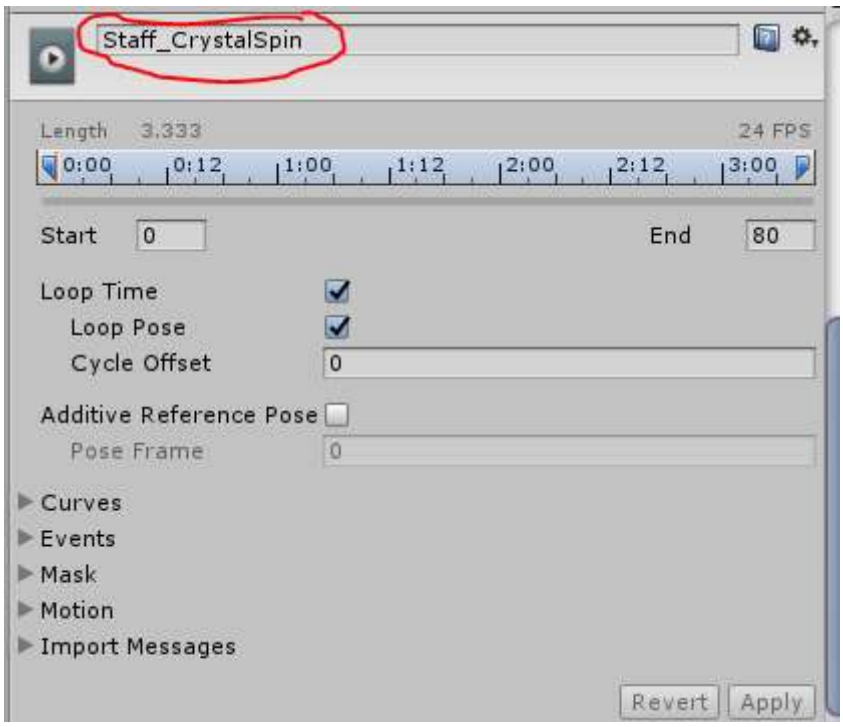
If you try to test your animation in Unity (or you finish building your item and test in-game) you will see that your Accessory's rotation is changed.

What happened? While I only animated the Spinner bone, there are animation keyframes for the other bones and the item's root. The animation on the item's root has the rotation saved, so our zeroing out the rotation got undone by the animation. Also, the animation on N_move screws up our ability to adjust the accessory in-game. You can see where all these new keyframes showed up by looking at the Animation Clip in the Animation window.

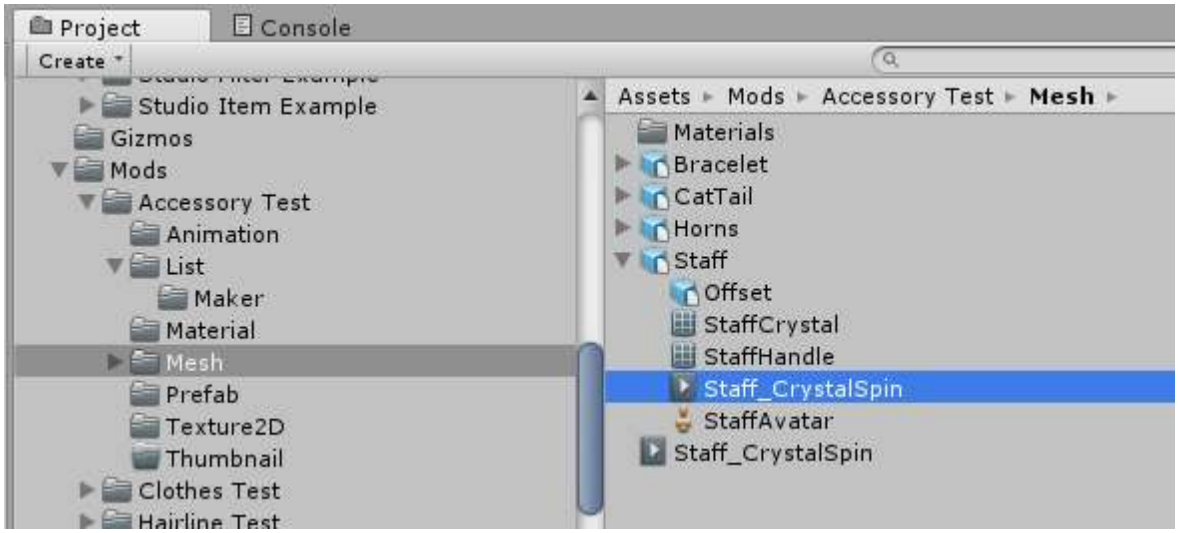


This also happens when you make a self-animated Studio Item, but it didn't cause problems for a couple reasons that aren't worth getting into.

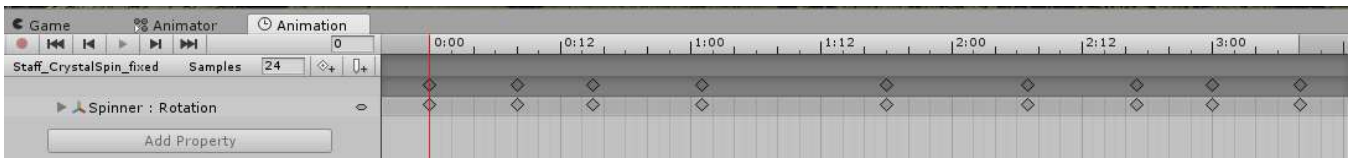
As far as I know you can't stop Unity from creating these extra keyframes, and you can't edit an imported Animation Clip. So what we want to do is duplicate the Animation Clip so we can edit the copy, but you will get an error if you try to duplicate it because the animation name has a "|" (vertical bar) in it. So in the import settings rename the Animation Clip first. Don't forget to Apply.



Then you can duplicate the Animation Clip (Ctrl +D).



Rename the new Animation Clip to avoid confusion, and then in the Animation window delete all the extra keyframes for things you didn't animate. Note that these remaining keyframes are not actually my original Blender keyframes; there's some other animation shenanigans you can look into if you want, but they're not critical.



Now you just need to replace the state's Animation Clip on your Animator Controller and you can finish making the mod as usual (make prefab, assign to AB, list file, etc.).

Self-Animated Accessory: Result

Now everything should be properly placed and animating. No more problems with N_move.