3D Items Part 1 - Studio Item Guide

Wogrim's Epic Guide to Creating Studio Items with Blender and KK Modding Tools

Before Reading This Guide

We are here to make Studio Items out of something you have already brought into Blender or created yourself. I may mention some tricks in some areas, but this guide is not made to teach much/any of the following because they are general things you could learn elsewhere and would make the guide too long:

- general Blender skills you can learn on YouTube
    - import models from outside sources into Blender
    - model or sculpt in Blender
    - UV unwrap
    - create textures for your item
    - rigging
    - skinning
- general Unity knowledge
    - GameObjects, Components, prefabs, materials
- basic modding stuff that you can learn from lower level guides
    - common terminology
    - basic structure of a zipmod
    - what shader to use / how it works / what textures you need
    - how to install Unity or the KK Modding Tools project

What Are We Making?

We are making a couple different types of items that you can place in Studio

- unskinned Studio Item
- skinned Studio Item - dynamic bones / FK posable

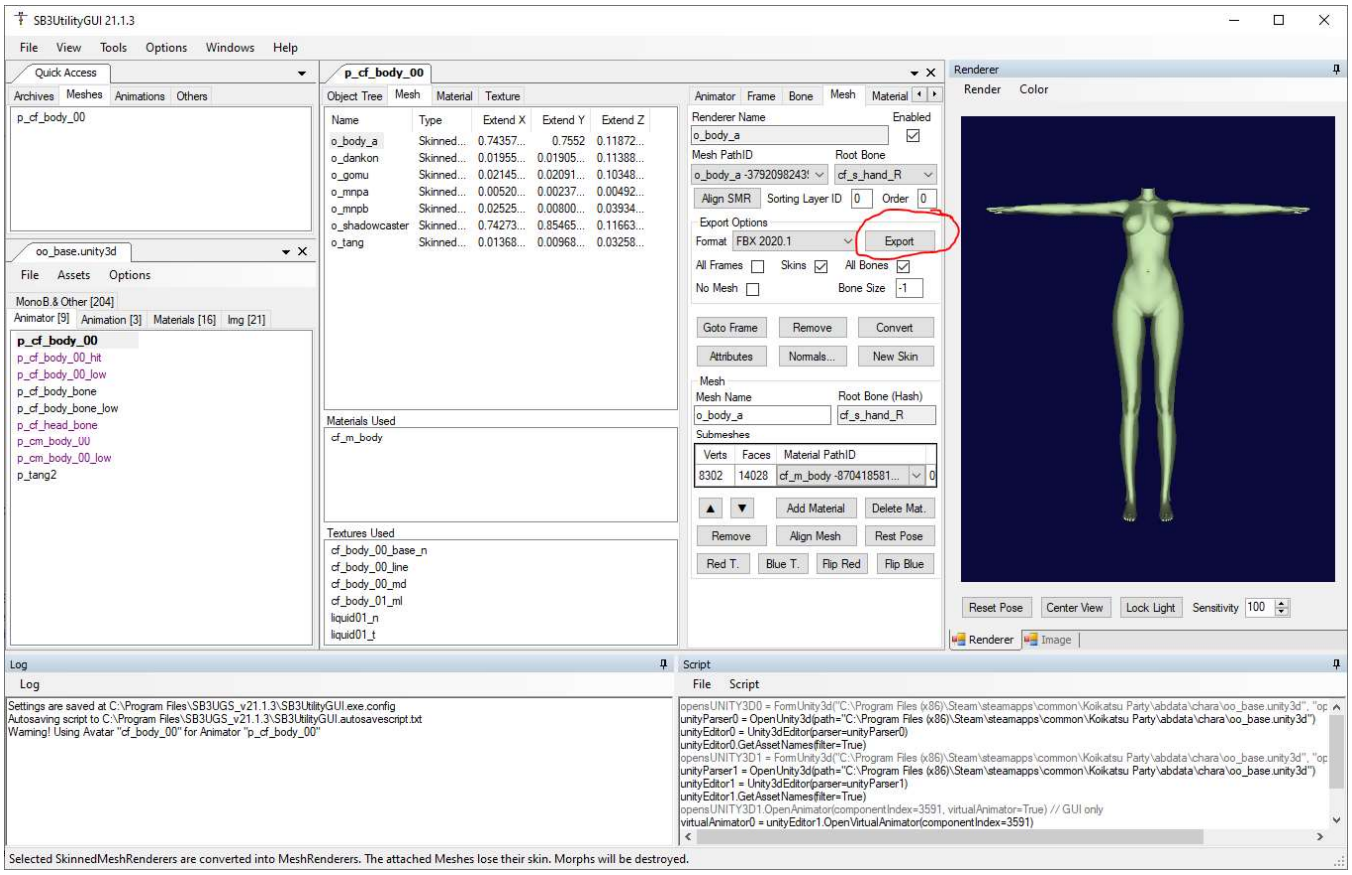
You may be tempted to skip the unskinned Studio Item part of this guide, but the steps that apply to all Studio Items will be described there, so you will miss important information.

If you don't have a Studio Item you want to make, it is recommended you export a game mesh (with SB3U) so you can get hands-on experience. The game's studio items are directly in the **abdata/studio** folder, but the materials (and textures) are located in a different AB under **abdata/studio/mat**. You can also just look at the items to study how they work.

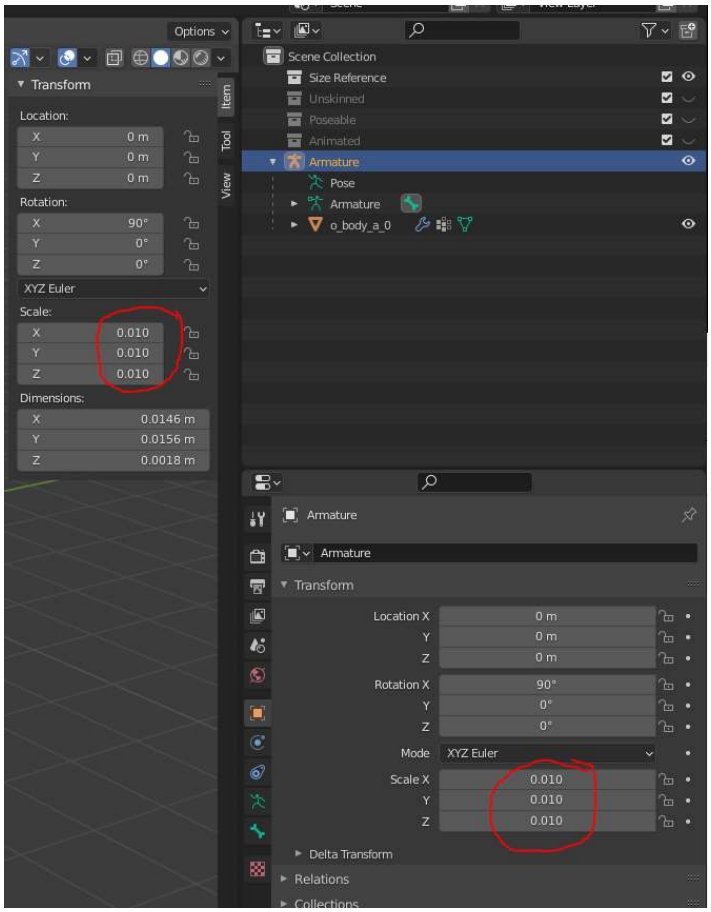
Preparing Blender

I first create a Blender project that has the KK body for size reference. I will be using the same Blender project for all 3 items, but you could make them separate if you want. I have created a new .blend file and deleted the default things. In the Outliner I have created new Collections and renamed them so I can easily hide and show things later. This can also be used to export your FBX as "Active Collection" instead of "Selected Objects".

So to get the body open **abdata/chara/oo_base.unity3d** with SB3UGS, double click on **p_cf_body_00**, go to the Mesh tab and click **o_body_a**. Export with the default FBX options.



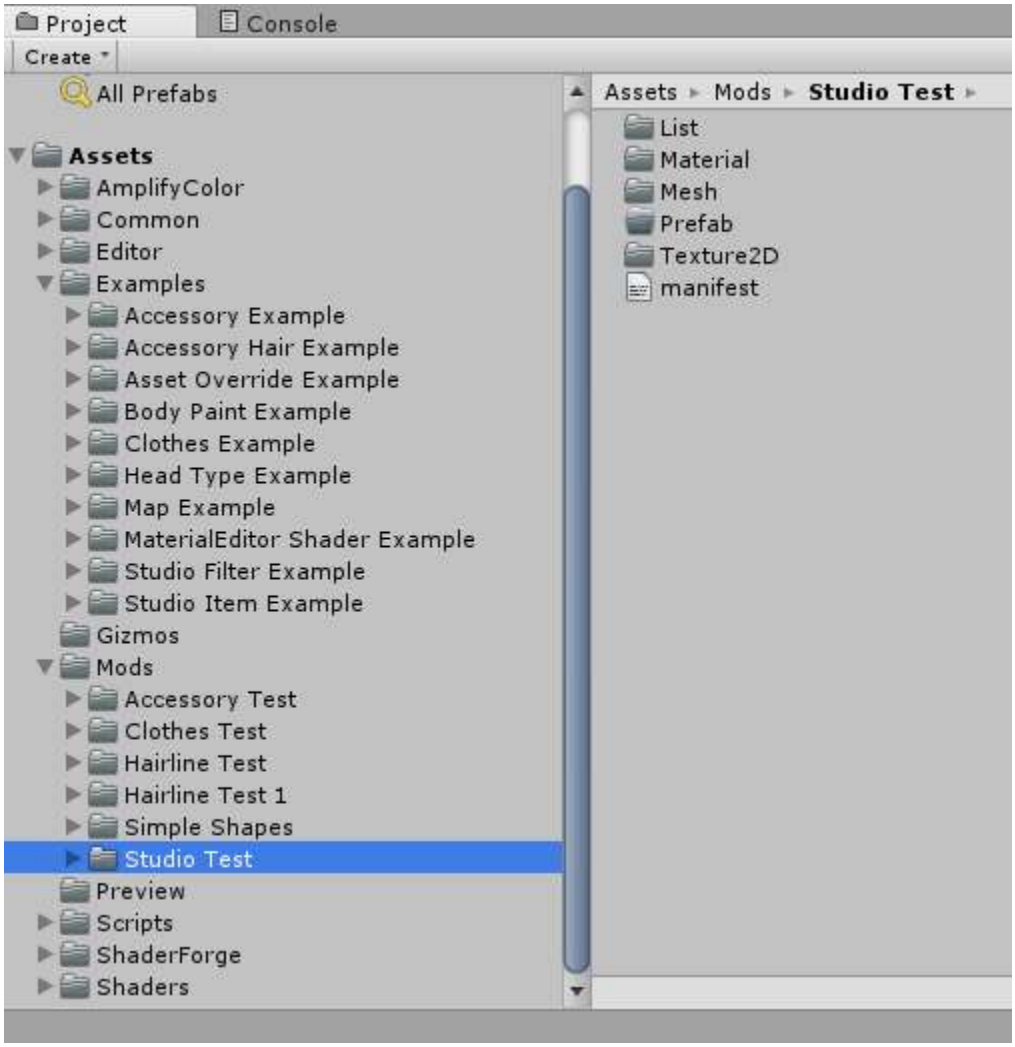
This will create an FBX file in **abdata/chara/oo_base/p_cf_body_00** but you should move the new folders somewhere else to not clutter your game install. In Blender, import the FBX. I do not change any of the import settings. The outliner should now show an Armature and a mesh inside it, but it is too small to see in the 3D Viewport. So select the Armature and you can see in the Object Properties or the 3D Viewport's Sidebar (N) that the scale is 0.01; change those to 1 and the body will become normal size. You may notice there is 90 degree rotation on X axis: this is because in Unity Y axis is up, but in Blender Z axis is up, so the item is rotated to look right; don't worry about it for now.



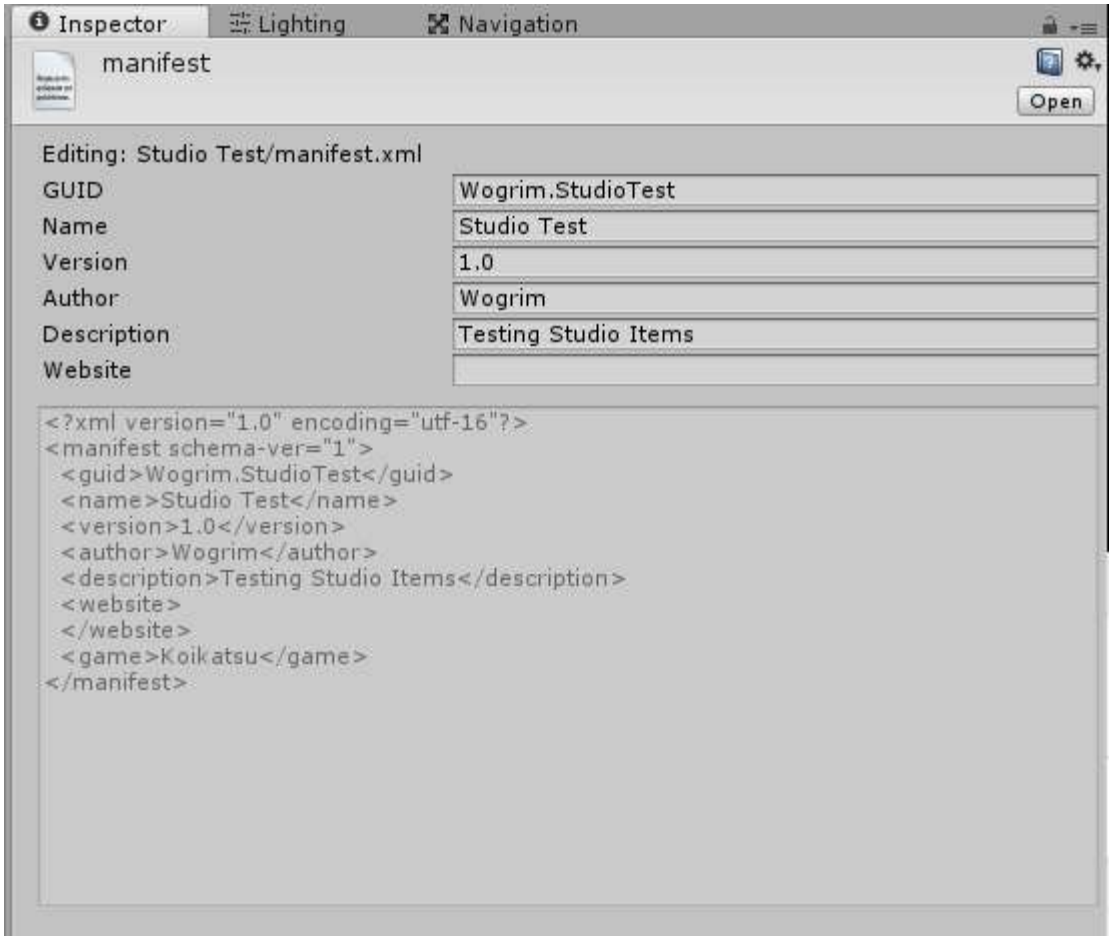
Move the armature and mesh to the Size Reference collection and move it a bit off to the side so when you have your item in the middle, you are able to compare the size easily. Hide the armature so you can see the body mesh without the bones.

Preparing KK Modding Tools

So open up your KK Modding Tools project and duplicate (Ctrl+D) the Studio Item Example folder. Move it into the Mods folder and rename it something appropriate for your mod. I've called mine Studio Test.

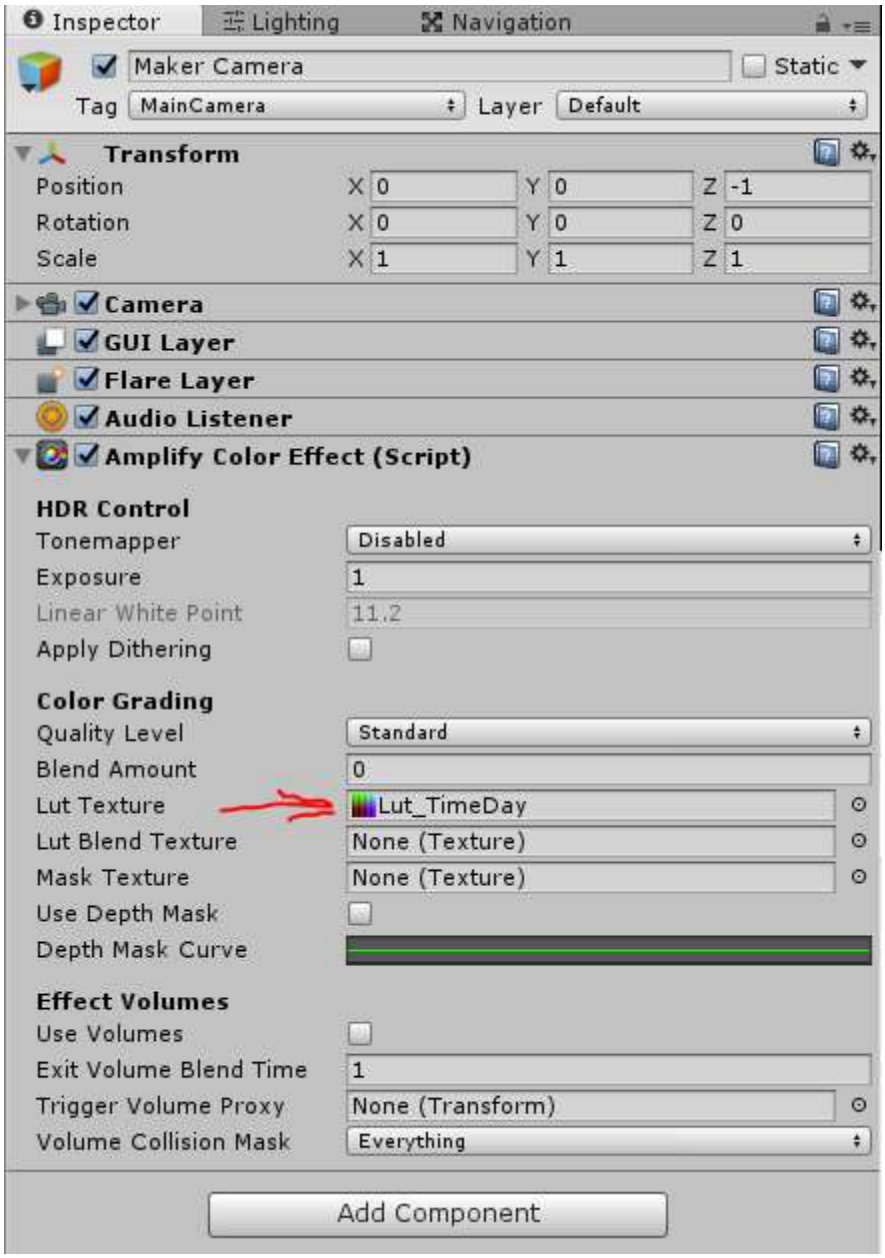


Single-click the manifest and you'll get a special editor (thanks to KK Modding Tools) in the Inspector window for filling it out, or you can open it and edit it normally.

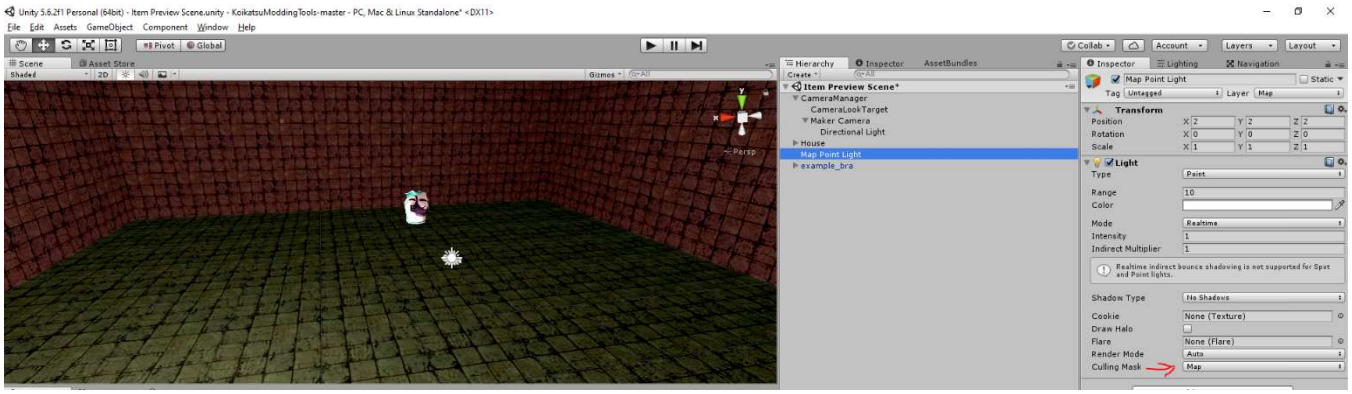


Now delete everything in your mod's folder except the manifest, the folders, and the list files.

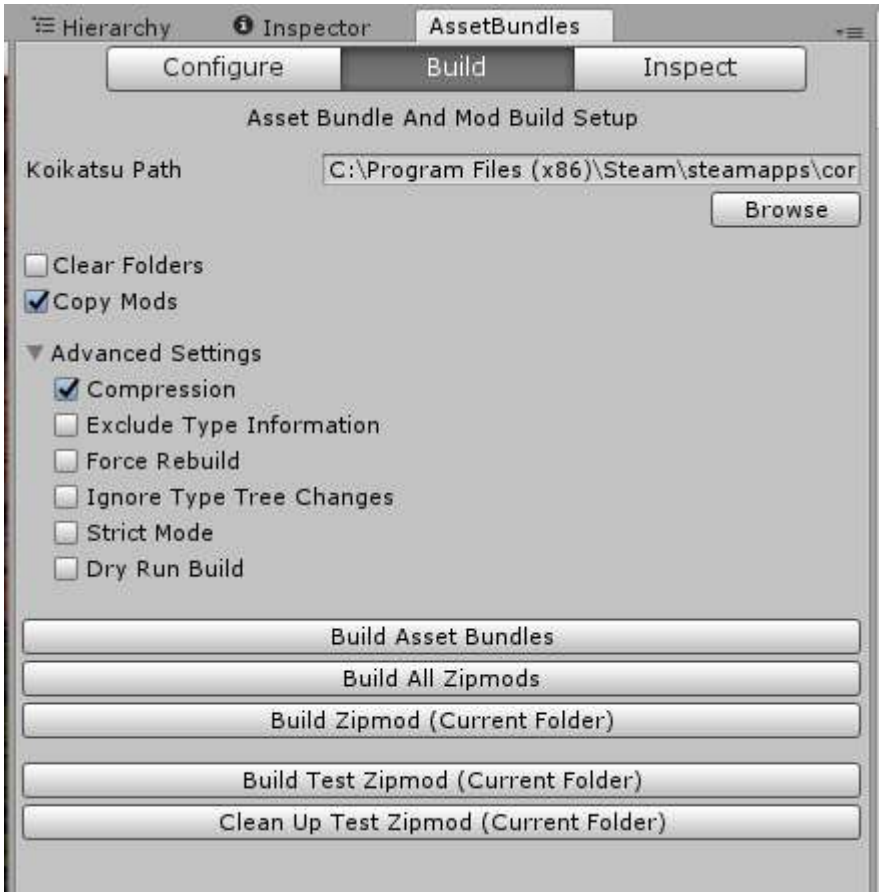
Now, if you do not already have the Item Preview Scene open, find it in the Assets folder and open it. Select the camera in the Hierarchy window, and you should be able to see a Amplify Color Effect component on it. You want to drag one of the LUTs from Assets/Common/LUT into the "Lut Texture" slot so that you can see things with the corresponding Studio Filter on the item.



You can add extra things to the scene, but if you don't know how, you will have to look for a basic Unity guide. I created a room with some textures so that when I look at things I have a nicer background and size reference. The room is on the Map layer so that it isn't lit by the Chara lighting attached to the camera; it is lit by a point light that only lights the Map layer (regular Unity shaders on the room). But if you want your item on the Map layer, depending on the shader you might need a directional light to see it properly. To limit what layers the light affects, you set the "Culling Mask". I can't remember if the directional light attached to the camera only affects the Chara layer by default, or if I changed it do that because it didn't look good on the room.



If you don't have the AssetBundles window docked somewhere, find it in the main menu under Window -> AssetBundle Browser and drag the tab somewhere to dock it. Fill out your game install path and make sure Copy Mods and Compression are checked. This is a KK Modding Tools thing that makes the final steps of building a mod super quick. You can Build Asset Bundles now and it'll build all the ABs for all the examples, otherwise it'll do it later when you just want it to build your AB.

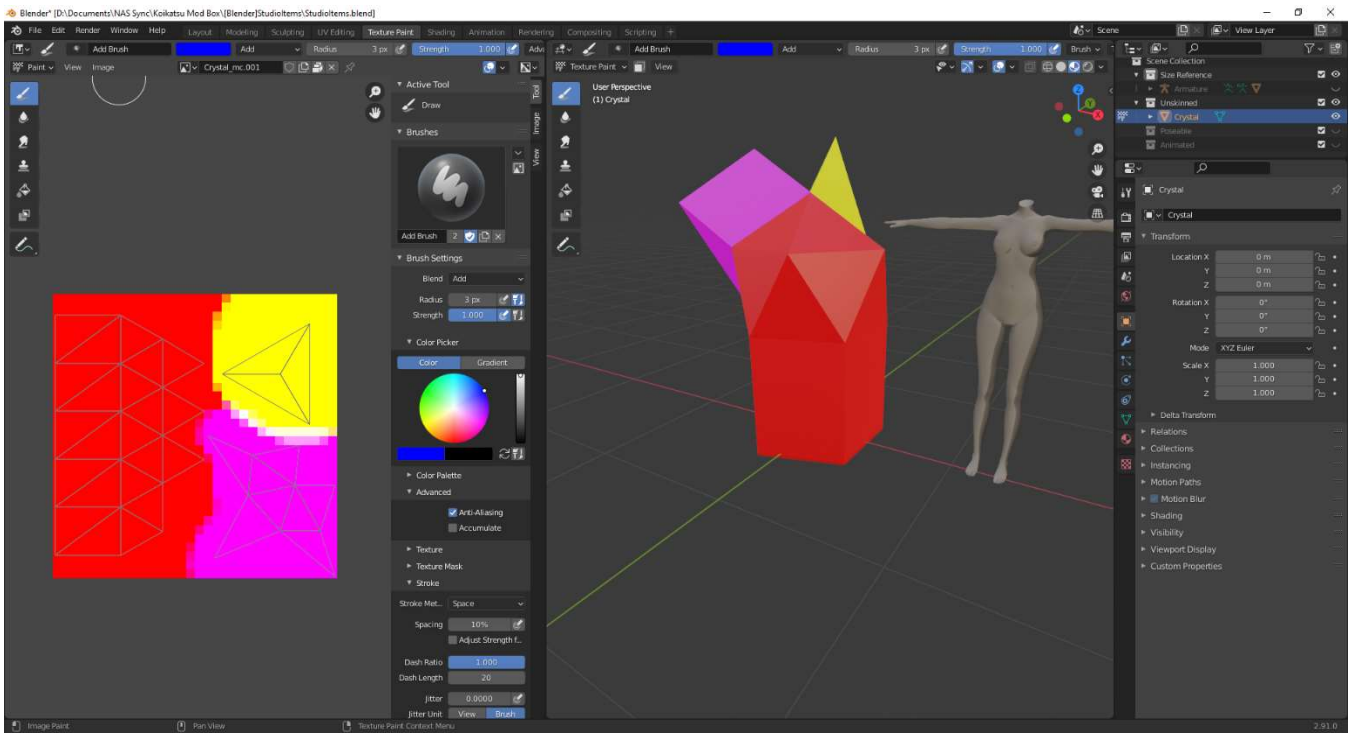


One last note, do not save scene or save project with an item with game shaders in the scene, it will cause problems on your material.

Unskinned Studio Item: Blender

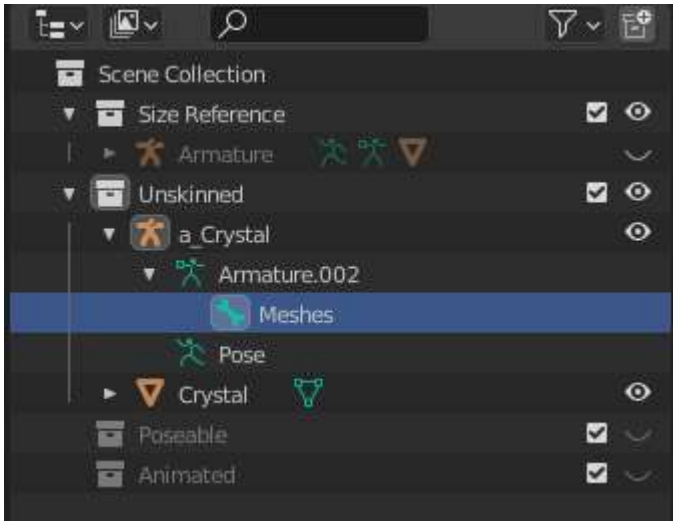
So now import or create yourself an unskinned item in Blender. So just a mesh (or multiple meshes is fine). If you don't know modeling or UV unwrapping or texture painting, find a YouTube video. Put it in its own Collection. If you import a skinned item you want to be unskinned, you can make it unskinned by removing the armature modifier on the mesh. If you import an unskinned mesh it may still come with an armature as its parent; to follow along properly, clear the parent on the mesh and delete the armature. If your mesh has any location, rotation, or scaling, Apply All Transforms after you have it how you want it..

So I've created this low poly crystal thing about as tall as a person. I UV unwrapped it in a way to keep separate colored areas apart. I made a 8x8 white maintex, a 32x32 color mask (shown here) and a 256x256 detail mask I texture painted a light shadowing on, not sure if it will be noticable. This is going to be an opaque crystal, not transparent.



So now what we're going to do is build a mini hierarchy for our item. You can create an unskinned Studio Item without doing this, but you will need to do it for more complicated items so you might as well learn it now. There are multiple ways to do this, and sometimes you can fix it in Unity, but I'd rather get it setup here. I will show a very simple hierarchy, but if you want to make the same item into an Accessory you should make the Accessory hierarchy so you can make both item types from the same FBX (I won't explain the Accessory hierarchy in this guide though).

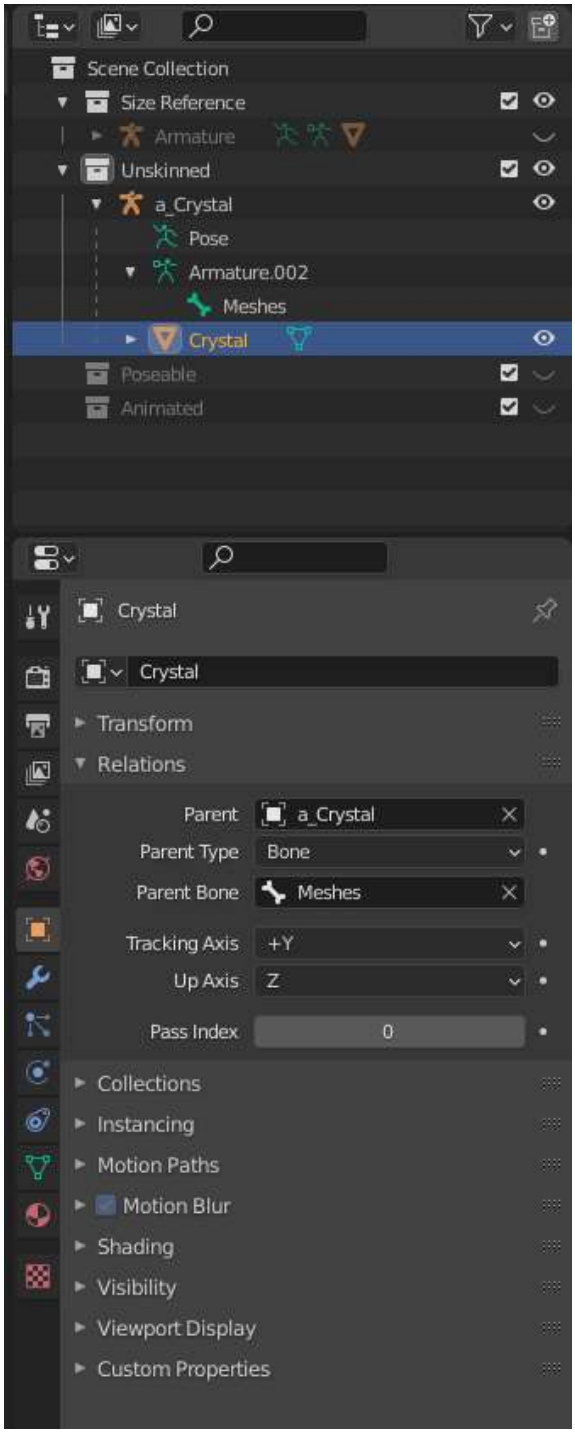
So in the same Collection as your mesh, create an Armature. I rename it something more identifiable, and rename the bone that gets created with it.



So now I want to set the "Meshes" bone as the parent of my mesh. There may be another way to do this, but here's the steps I use:

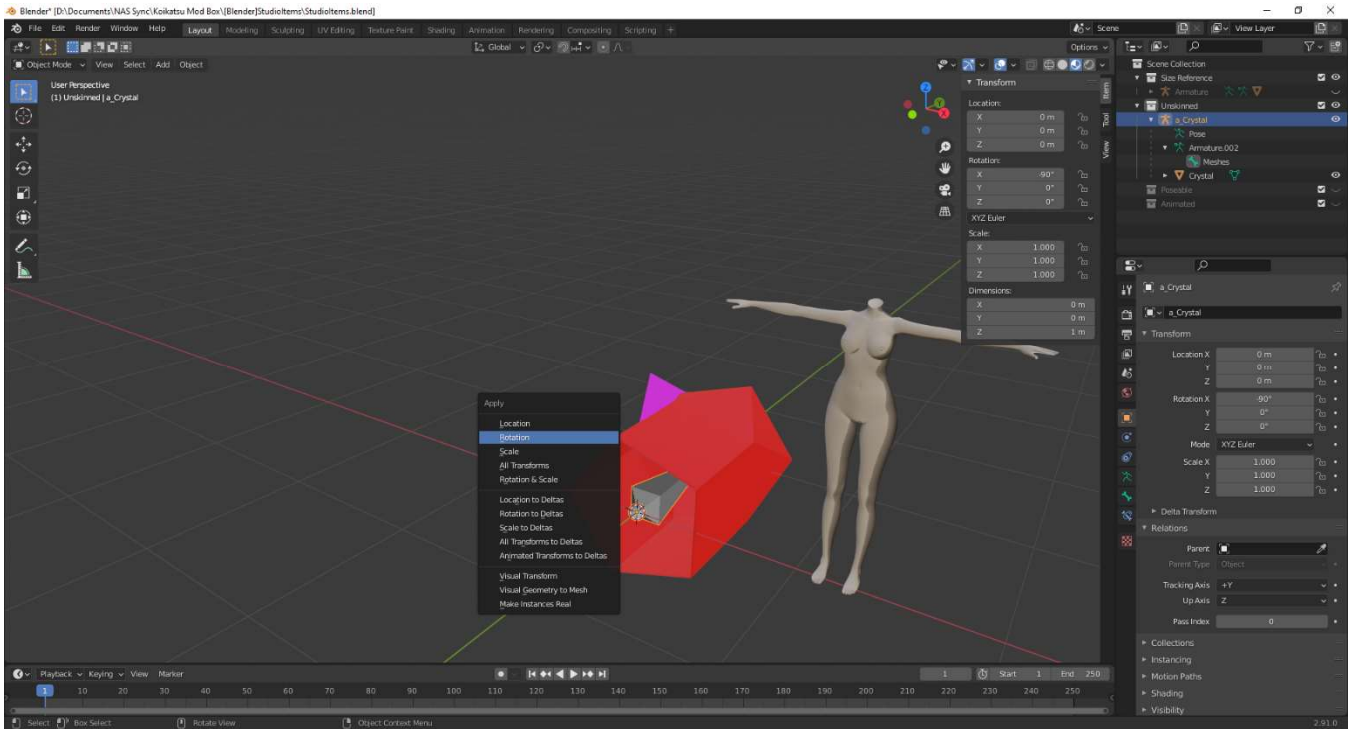
1. select armature in object mode
2. Tab into edit mode
3. select bone that you want to be the parent
4. Ctrl + select the mesh that will be the child
5. Tab into object mode
6. Ctrl + P -> parent to Bone

The mesh will show under the armature in the Outliner, but if you look at Relations in the Object Properties you will see the bone is the parent.



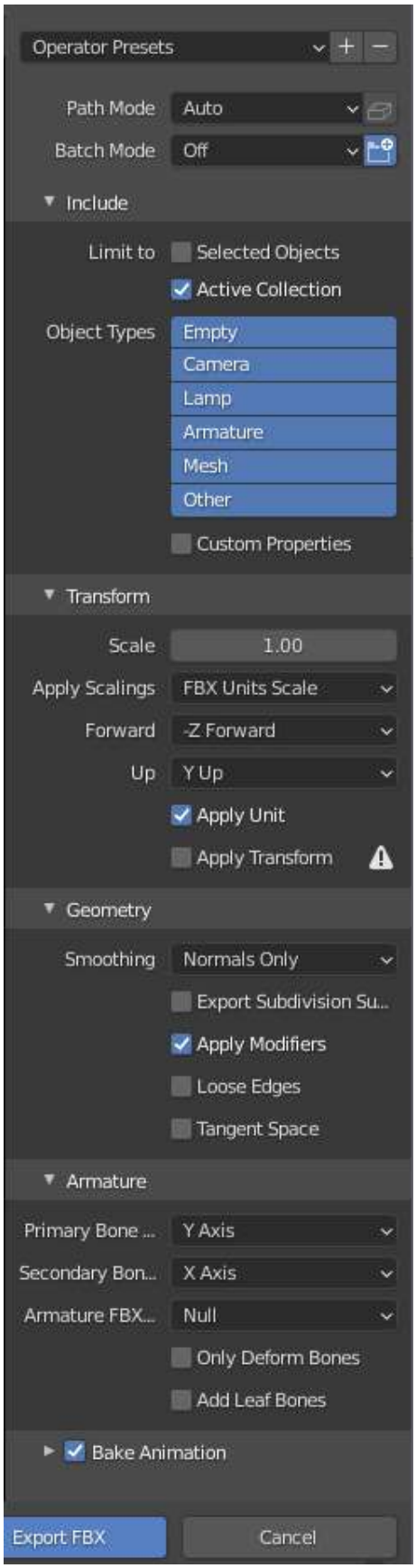
So now we have one more step before we export from Blender, which is to fix the rotation. I mentioned earlier the body has 90 degree X rotation because of how in Blender Z is up but in Unity Y is up. We need our item rotated the same so that the rotation applied by the export cancels out and our item has 0 rotation in Unity (and is facing the right direction).

So to do this, select the armature in Object mode, rotate to -90 degrees on X, and apply the rotation.



Then rotate it to 90 degrees on X so it is standing back up and you are ready to export.

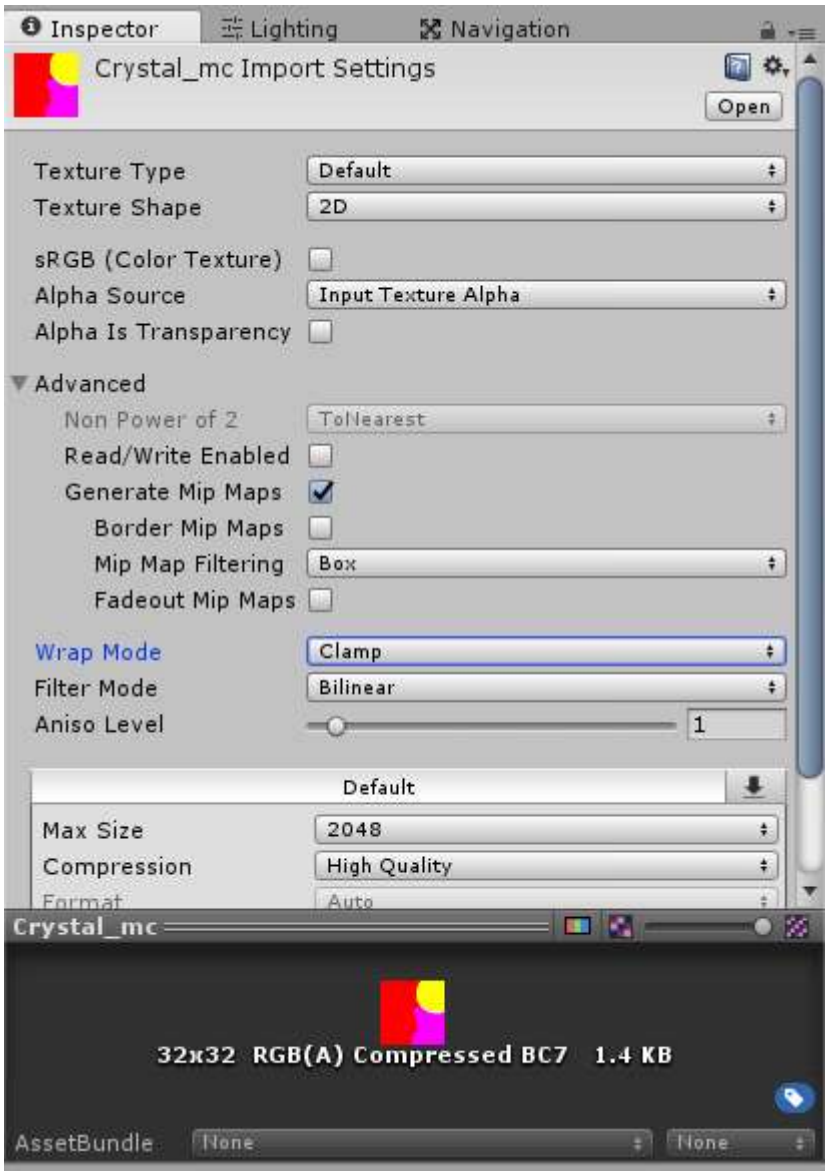
So export FBX. You can export selected (must have armature and mesh selected) or export collection (must have the collection or something in it selected. I'm not going to go into an explanation of these settings, just make yours look the same, and add yourself a preset for easy future use.


Operator Presets
Path Mode
Auto
Batch Mode
Off
Include
Limit to
Selected Objects
Active Collection
Object Types
Empty
Camera
Lamp
Armature
Mesh
Other
Custom Properties
Transform
Scale
1.00
Apply Scalings
FBX Units Scale
Forward
-Z Forward
Up
Y Up
Apply Unit
Apply Transform
Geometry
Smoothing
Normals Only
Export Subdivision Su...
Apply Modifiers
Loose Edges
Tangent Space
Armature
Primary Bone ...
Y Axis
Secondary Bon...
X Axis
Armature FBX...
Null
Only Deform Bones
Add Leaf Bones
Bake Animation
Export FBX
Cancel

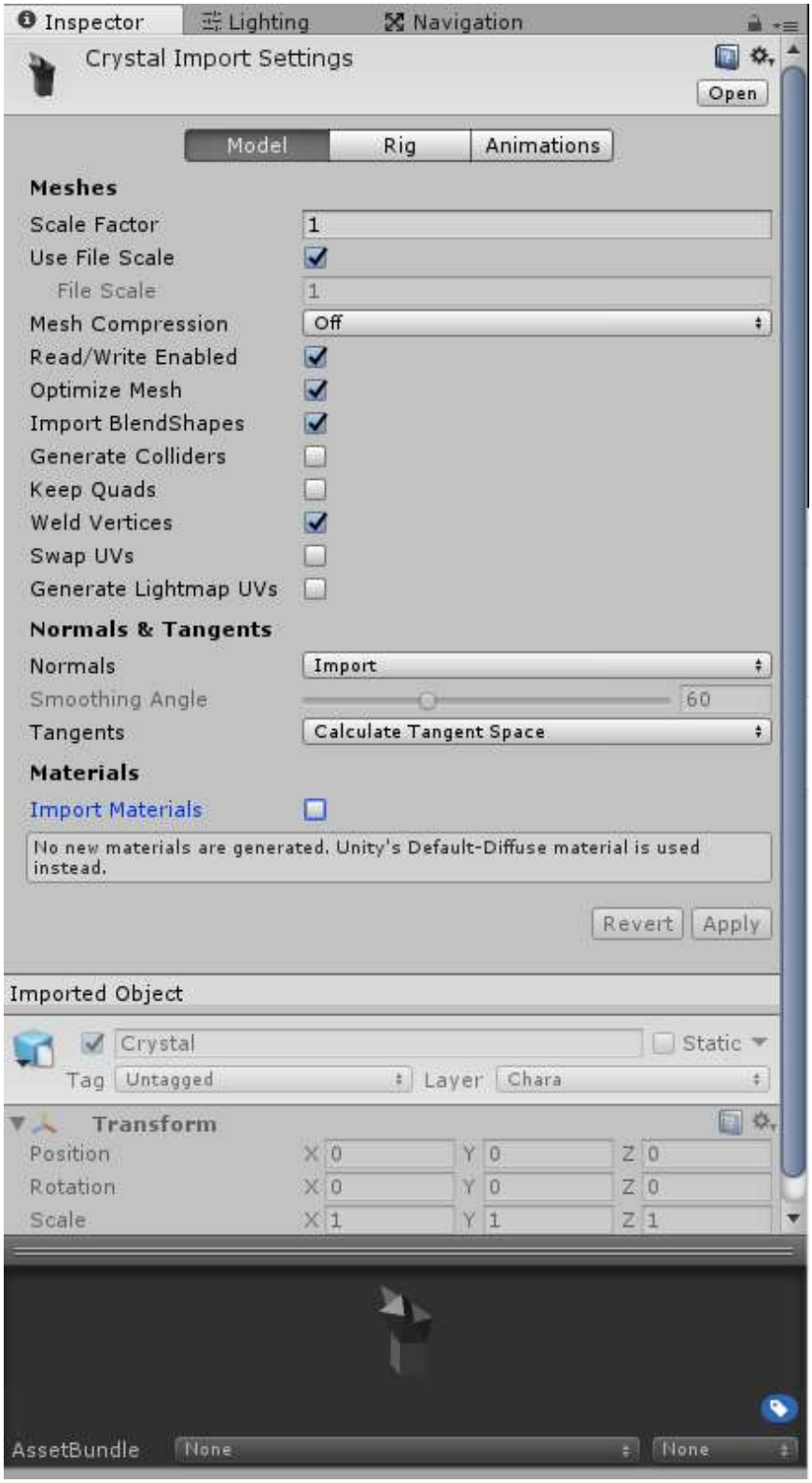
Unskinned Studio Item: Into Unity

Go into your Texture2D folder and import (drag and drop from Explorer) all your item's textures. If you created them in Blender like I did, you have to save a copy as a png first (use 0 compression). You may need to change import settings. Make sure to Apply any changes you make.

Here's what I did for my color mask. I unchecked sRGB because it is a linear texture. I let it generate mip maps. I changed wrap mode to Clamp because even though it isn't made to tile but shouldn't matter, the UVs are so close to the edge (and the resolution is low) that colors were bleeding over from the other side of the image. And I set Compression to High Quality for BC7 compression.

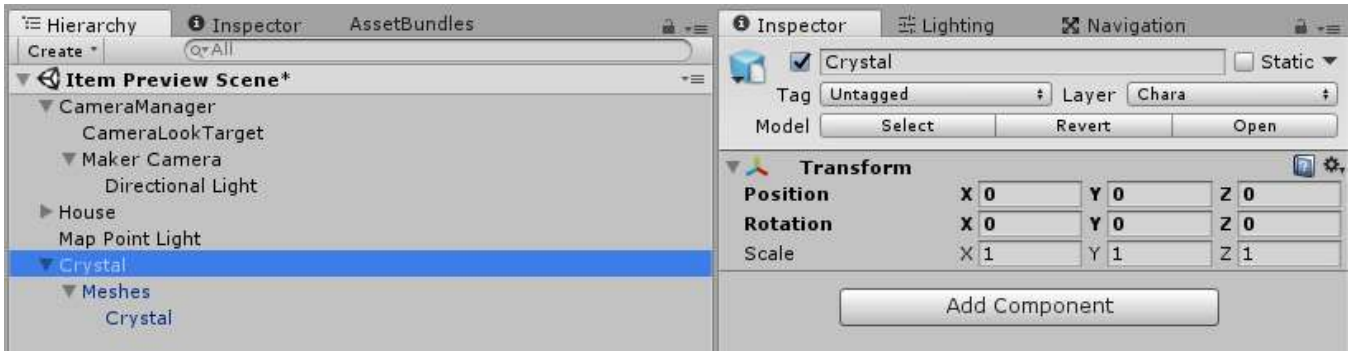


Now go to your Mesh folder and import your FBX. The only thing I change here is is I do not import materials. If you didn't match my Blender export settings, you may have a File Scale that is not 1. You may be able to fix this by unchecking "Use File Scale" but you may still have problems.



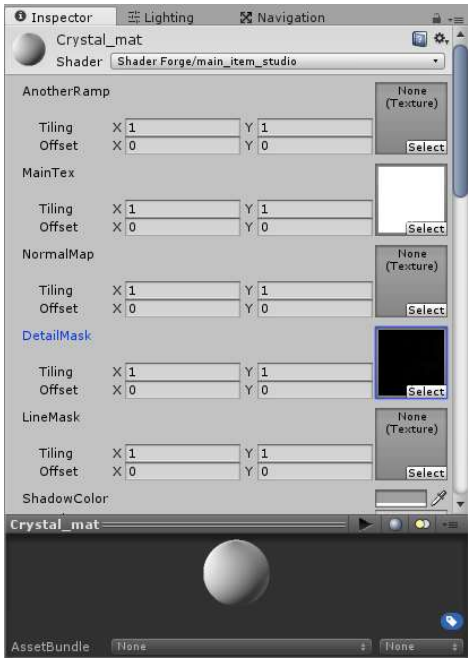
Unskinned Studio Item: Putting It Together

Drag the whole imported fbx thing into your preview scene's Hierarchy window, and look at the Transform of your item's root GameObject. You want Position to be 0,0,0 and Rotation to be 0,0,0 and Scale to be 1,1,1. If it it does not have these settings (called "identity" Transform) your item will probably be messed up when created in Studio. So working backwards, you have a problem on Unity import settings, Blender export settings, or your item's Transform in Blender before you exported it. Go back and fix it now, or finish making the item and see what happens.

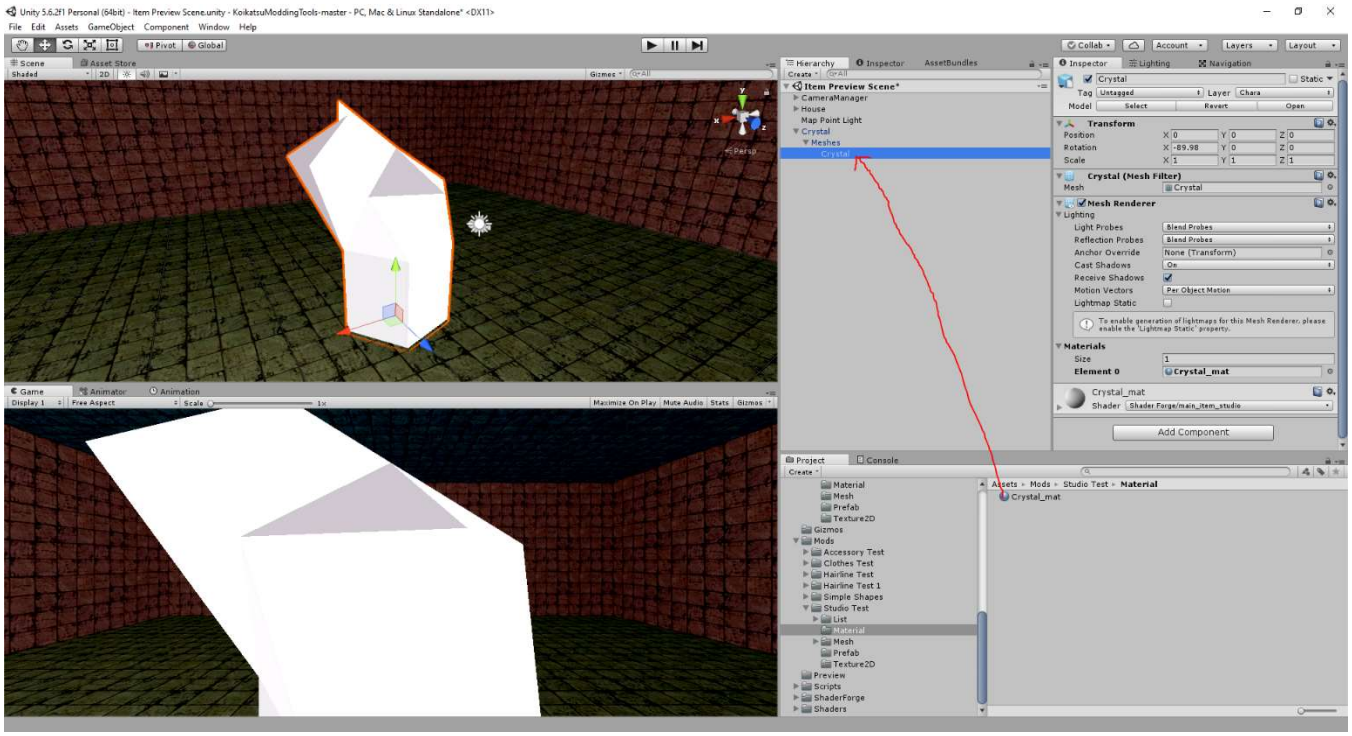


You also need to decide if your item will be on the Chara layer or the Map layer. It affects what lighting and shadows it will receive, so it is an important decision. KK Modding Tools automatically puts things in the Chara layer. If you decide to change it (upper right of Inspector window) make sure to do it on the root GameObject and apply to all children.

Now go back the Project window into your Material folder and right click -> Create -> Material. Name it something practical, and change it to an appropriate shader (Inspector window). If your item needs multiple materials, make them all. Fill in all your textures and some initial variable settings. You can figure out better settings later in Studio with Material Editor.

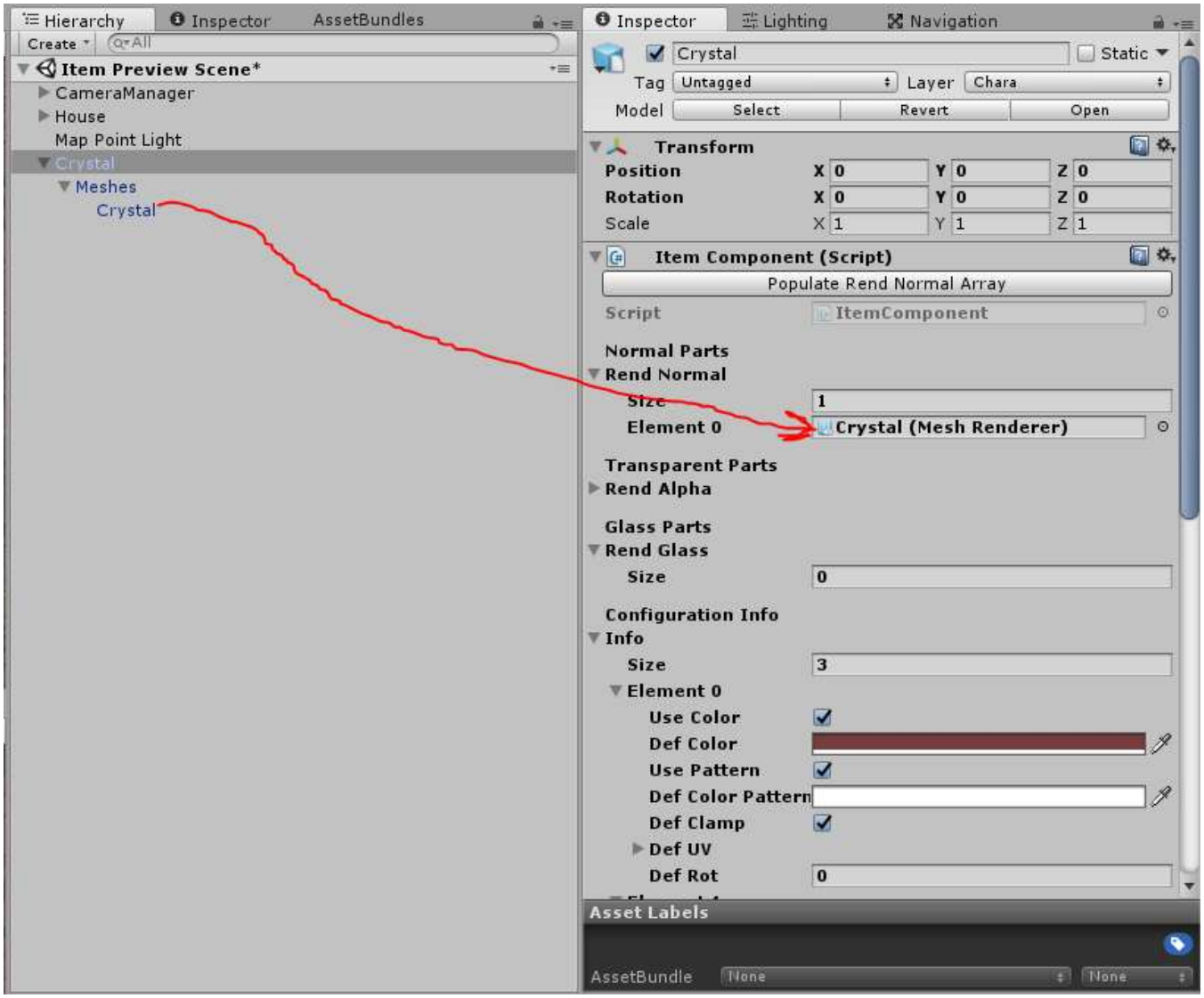


Drag the material onto the GameObject with the Mesh Renderer to give it the material. If you have multiple meshes, they each need a material (can be the same or different). Some of the game shaders will not look how you expect yet, do not be alarmed.

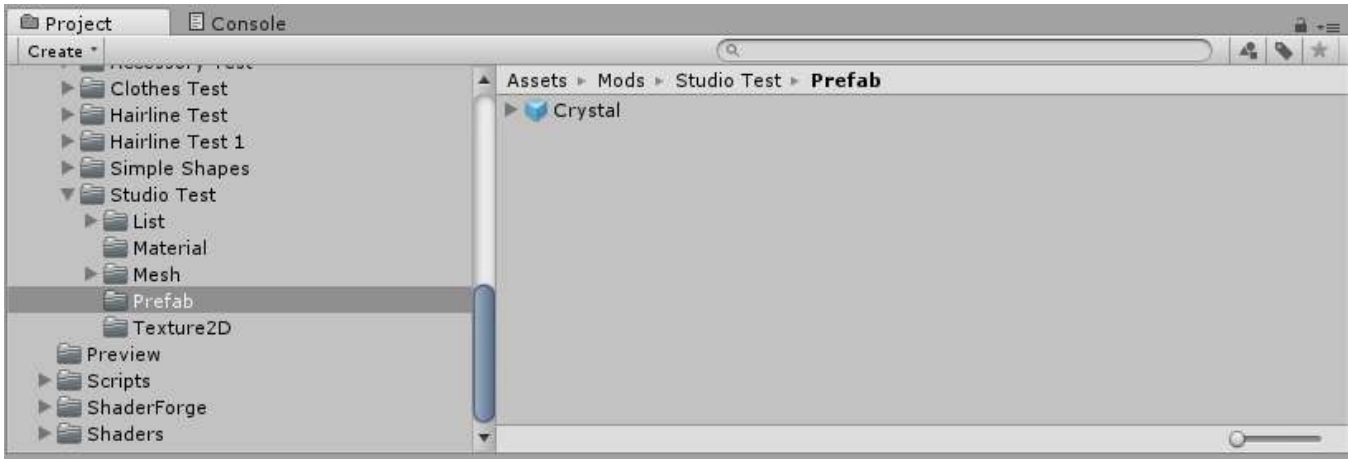


So the other thing our item still needs is the Studio Item MB, which is called "Item Component". Add it to the item's root GameObject and fill it out. Things you usually do for Studio Items:

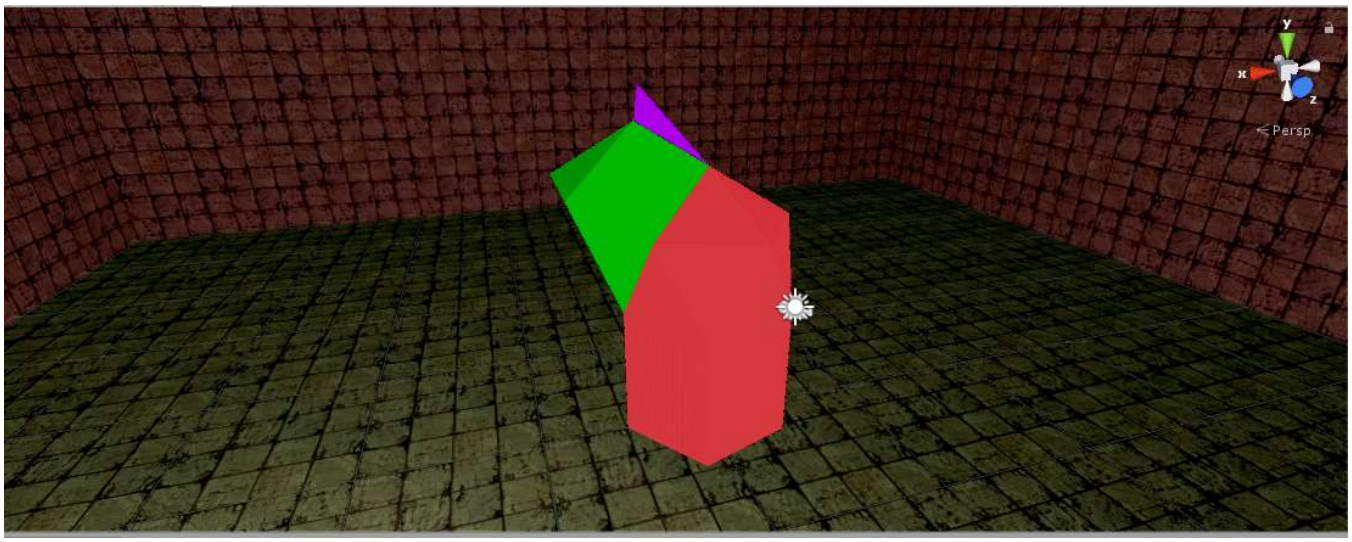
- set the size of the "Rend Normal" array to the number of meshes in the item, and then drag the GameObjects that have them from the Hierarchy window into those slots
- set up default colors ("Def Color") for each of the color picker slots
- check the "Use Color" buttons which will later give the item the default colors in the preview scene
- set up the default shadow color ("Def Shadow")



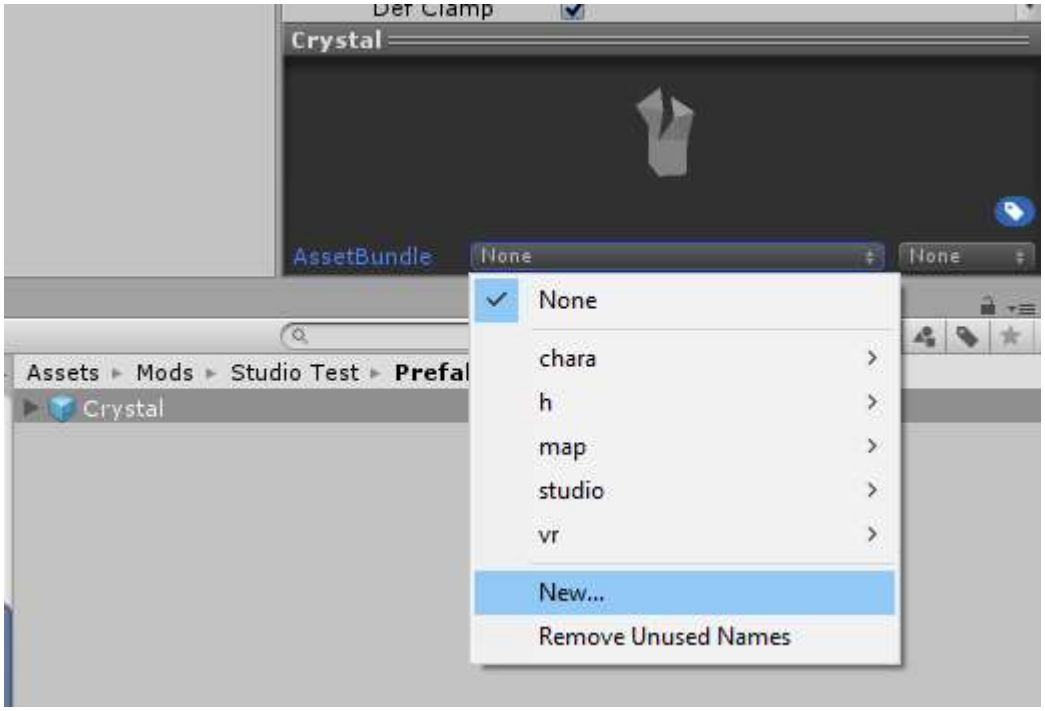
Now our item is ready to be made into a **prefab**, which is the form our item is in when we put it in the AB. To do this, just drag the item's root GameObject from the Hierarchy window into your Prefab folder (in the Project window). Delete the item from the Hierarchy.



Drag the prefab back into the Hierarchy window, which creates a prefab instance. If it is colorable, it should have the default colors you set up in the MB. This is pretty much how the item will look in game, so if you don't like how it looks now is a good time to change it. I don't think the light colors look good on my crystal but I will leave it as is so you can compare to a game screenshot later. Shadow color doesn't seem to work properly. I'll figure out better settings in Material Editor, then come back and change the default colors. If you make any changes on the prefab instance you want to save, hit the Apply button in the upper right of the Inspector window. Then delete the prefab instance.



We need to assign the prefab to an AB still, so select it (in the Project window) and at the bottom of the Inspector window, click the bar in the middle and give it a New AB. A good naming convention for this is **studio/author/modname.unity3d** so I call mine studio/wogrim/studio_test.unity3d all lowercase. This will put the whole hierarchy with the mesh and the material and all the material's textures into the AB, so the prefab is the only thing we have to manually assign to the AB. For other types of items you may have to manually assign other things.



Studio Item List Files

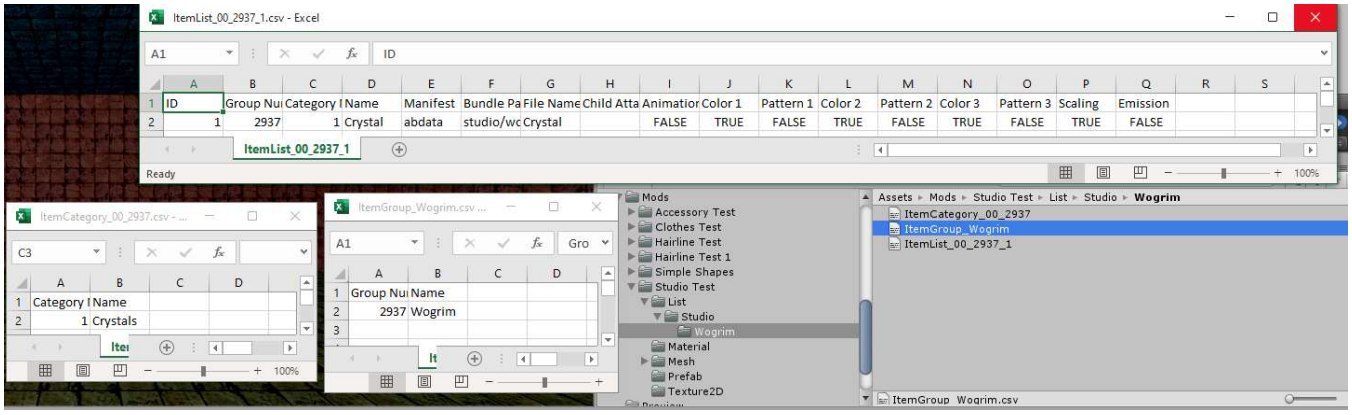
Studio Item list files are a bit different from the list files for other types of items. There are 3 different types of Studio Item list files:

- **ItemGroup** - this type of list creates folders in Studio under Add -> Items
- **ItemCategory** - this type of list creates folders in Studio under a specified Group
- **ItemList** - this lists your Studio Items within a specified Group and Category

You can see the game's list files for Studio Items under **abdata/studio/info** but looking at them isn't very helpful. Instead, look at the list files in the Studio Item Example in KK Modding Tools, and read **Studio Item Example/_readme.txt** for a good explanation. I don't know an easy way to see all the Group and Category numbers people are using for their mods, so if you're trying to make a new folder pick a random number and see if your stuff gets mixed in with someone else's mods. If you're trying to use someone else's Group or Category, you can try to find their mod in your mods folder, extract it and look at the list files.

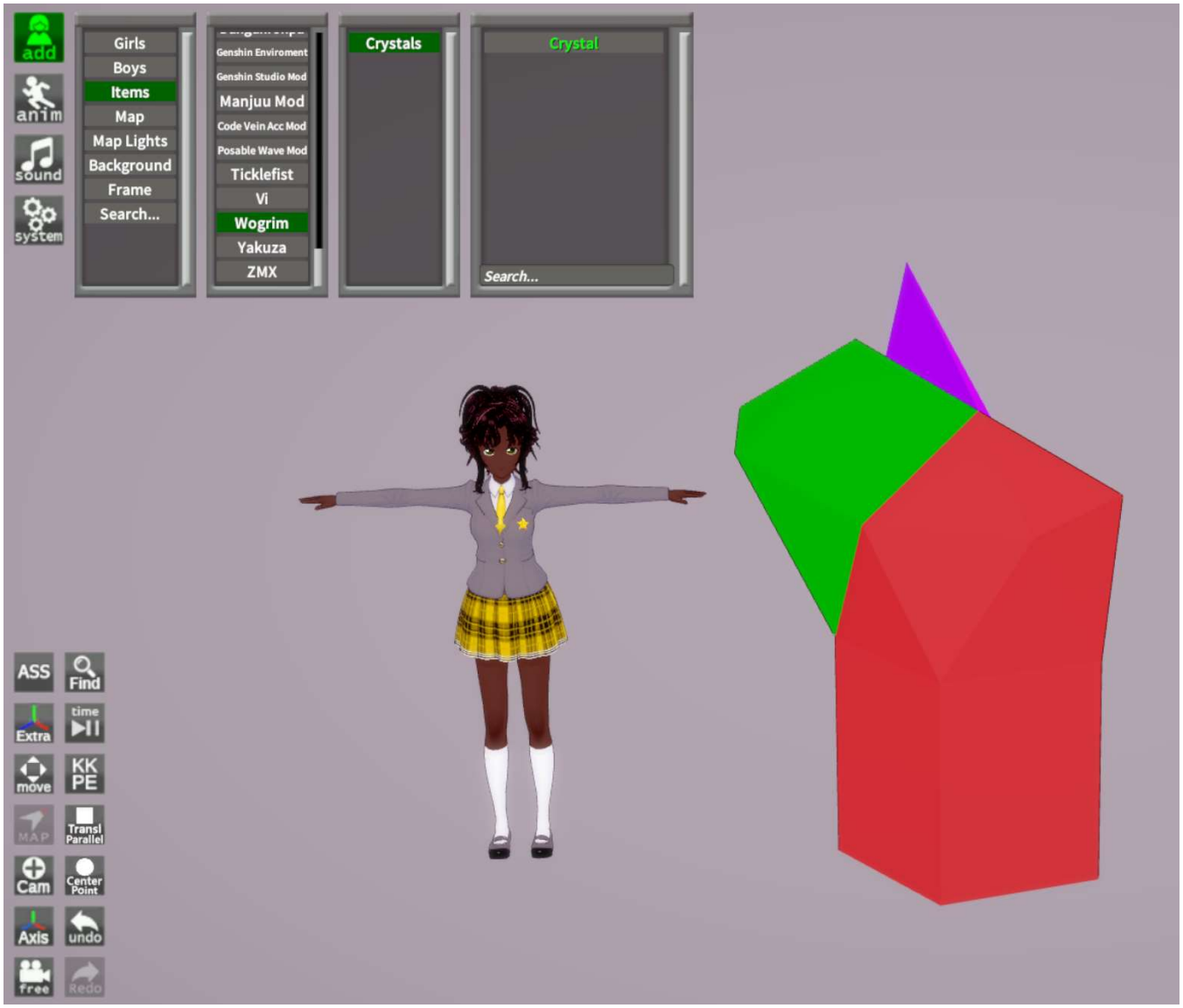
So I

- named my folder Wogrim, named the ItemGroup file to match
- picked a random number 2937 to be my group number, for a group named Wogrim, named ItemCategory file to match
- picked a number 1 to be category number (not worried about conflicts because it is my own group), with name Crystals, named ItemList file to match
- filled out item list file appropriately; File Name is the prefab name and Bundle Path is the AB you assigned it to



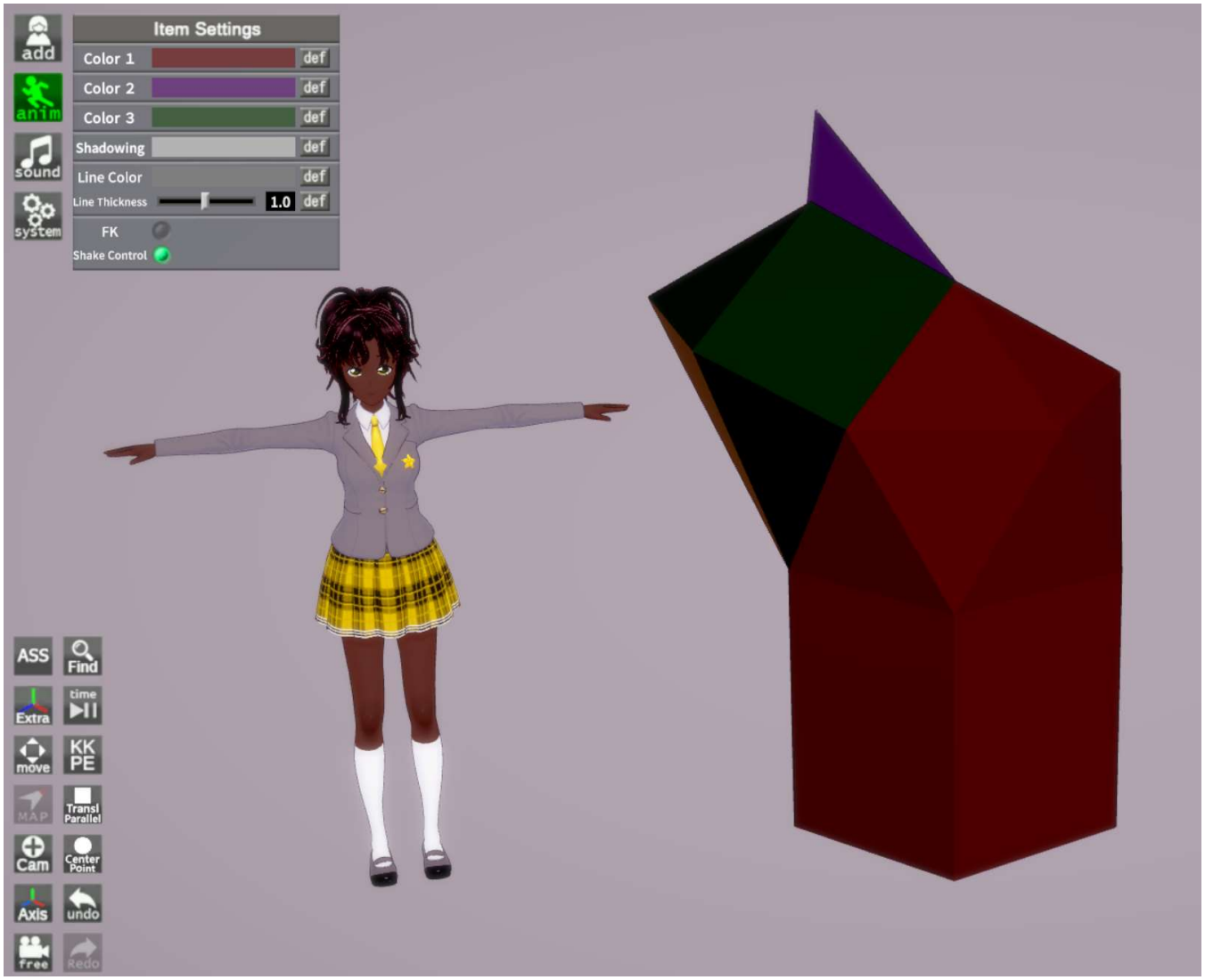
Unskinned Studio Item: Result

Now we just have to build the AB (we assigned the prefab to it before) and build the zipmod. If there's no errors (watch the Console window) this should just take 2 clicks in the AssetBundle window. One to click the "Build Asset Bundles" button, and one to click the "Build Zipmod (Current Folder)" button (make sure you have your mod's folder selected in Project window). Start up Studio and you should be able to find your item. If you can't find your group/category (but you can find it in quick access box) or there's someone else's stuff in it, you got a group / category number conflict and you need to try a different number. If you have other types of problems, grab your built mod from the mods folder and extract it somewhere so you can diagnose the problem.



First see that your item comes in at the expected size and rotation, then check that you can move, rotate, and scale it without problems. Then test other features like color picker. And finally play around with Material Editor to change settings if you're not getting the look you want.

So I darkened the shadow color a bit to give the item more definition, and chose darker color picker colors which helped with that and looks better with the rim lighting and just looks more realistic in general. I could have maybe got better results with a ramp texture, but I didn't want to mess with one.

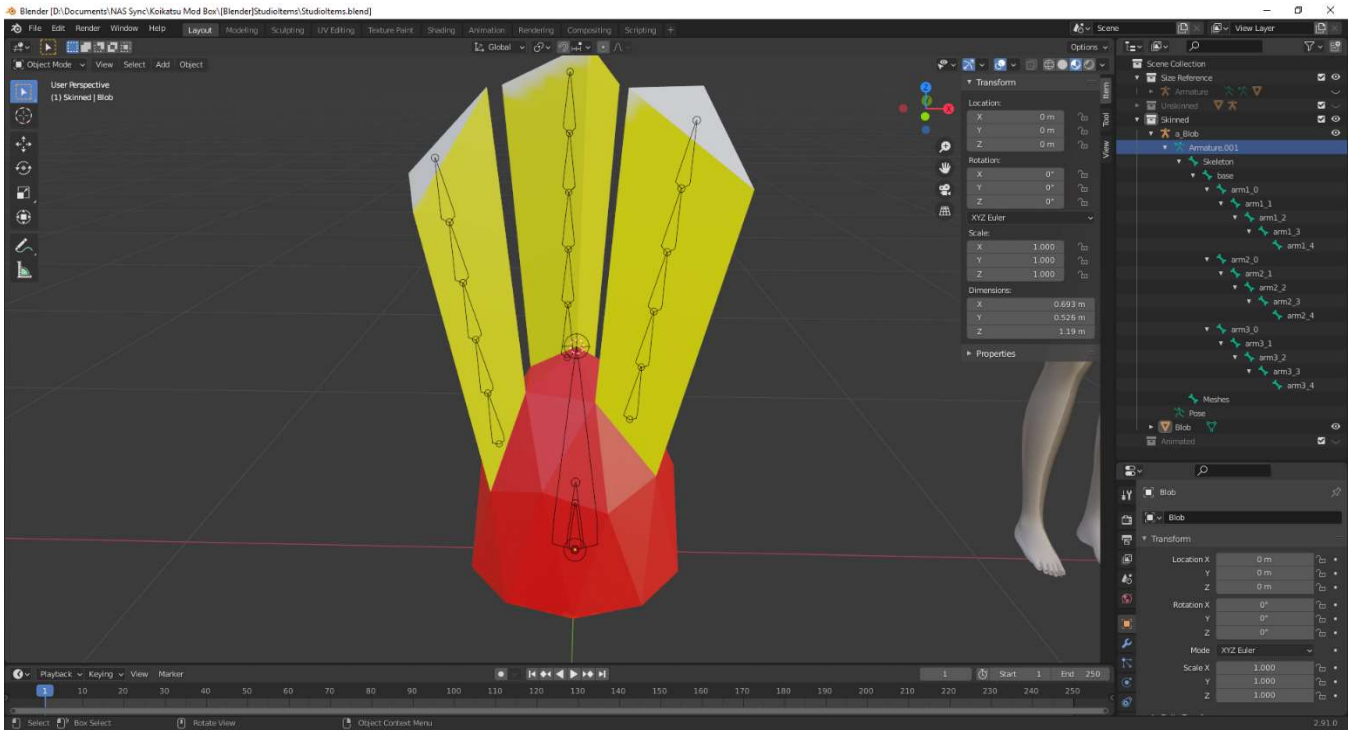


If you want to make a thumbnail for QuickAccessBox, look at the instructions on the github page for it. It requires putting a PNG file in a certain folder with a certain naming scheme. There isn't currently a way to do it within KK Modding Tools, so I'm leaving it out of this guide; you have to do it after you build the zipmod.

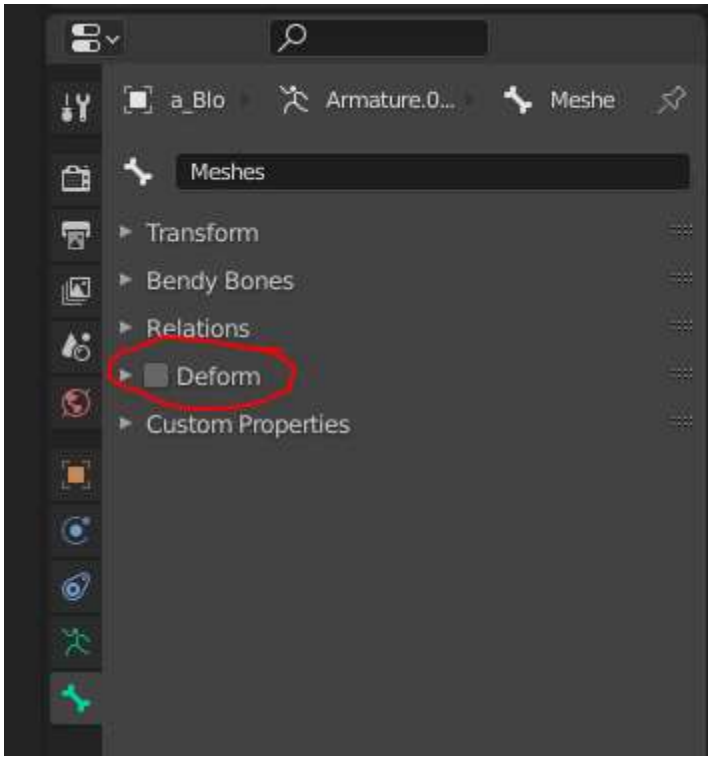
Skinned Studio Item: Blender

So it's back to Blender to create a skinned Studio Item. We will be making it posable with FK in Studio and also give it dynamic bones so that it'll move a bit if attached to something animated, but all we need to do as far as Blender goes is to make a skinned mesh and a proper hierarchy. Again, if you want to also make this item into an Accessory you'd do a little extra stuff; for this guide I'm trying to keep it as simple as possible. If you don't know how to do rigging and skinning, find a YouTube video.

So here I'm making this knee-high "Blob" with a few tentacles; I have modeled it, UV unwrapped it, created a main tex and color mask, and created the armature. At the top level of the armature, we have the "Meshes" bone again which will be the parent of the mesh, but we also have the "Skeleton" bone which is the parent of all the bones the mesh will deform to.



So I'm going to do automatic bone weights to make this quick, but the problem is if I do that as-is, the mesh will have bone weights for "Skeleton" and "Meshes" bones. So I first go to Bone Properties for them and uncheck "Deform".



I make sure the mesh and armature transforms are identity, then do automatic weights. This puts the armature as the parent of the mesh, so I still have to do the procedure to set the Meshes bone as the parent. Then I did some bone weight fixing so when the tentacles move it doesn't move the base. Last step before export is just as before, where I rotate armature to -90 on X, apply rotation, rotate to 90. Export textures and FBX same as before.

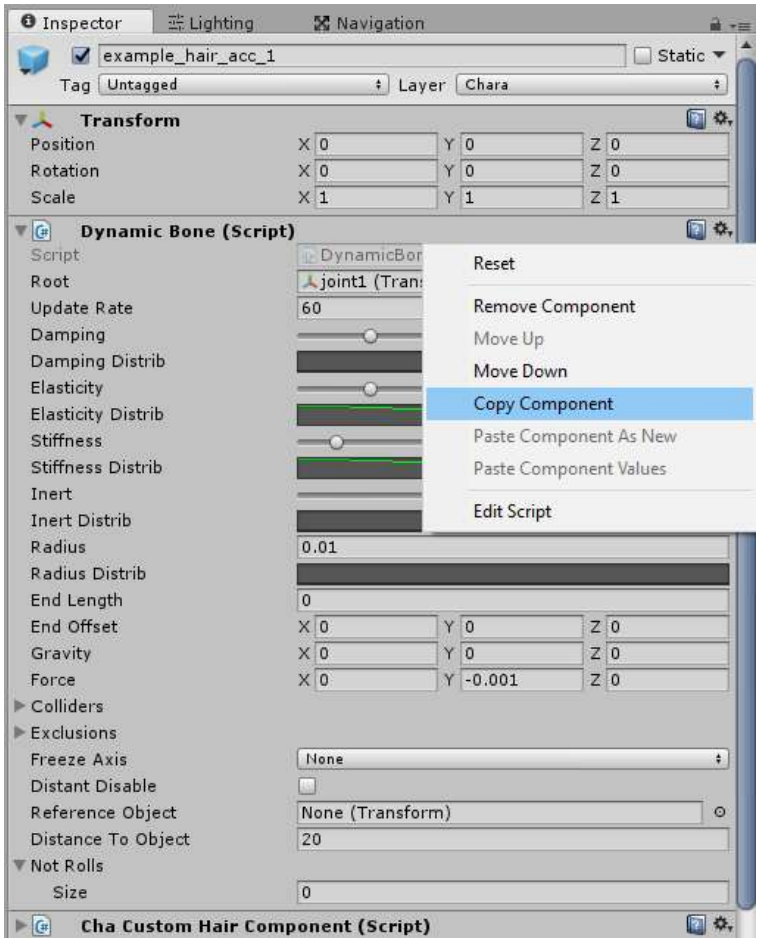
Skinned Studio Item: Unity

Import the FBX and textures to the same folders as before, same settings as before. Create your item's Material and put the textures on it, nothing new unless you're using a different shader or something this time around. Drag the imported FBX thing into the Hierarchy and make sure it has identity Transform.

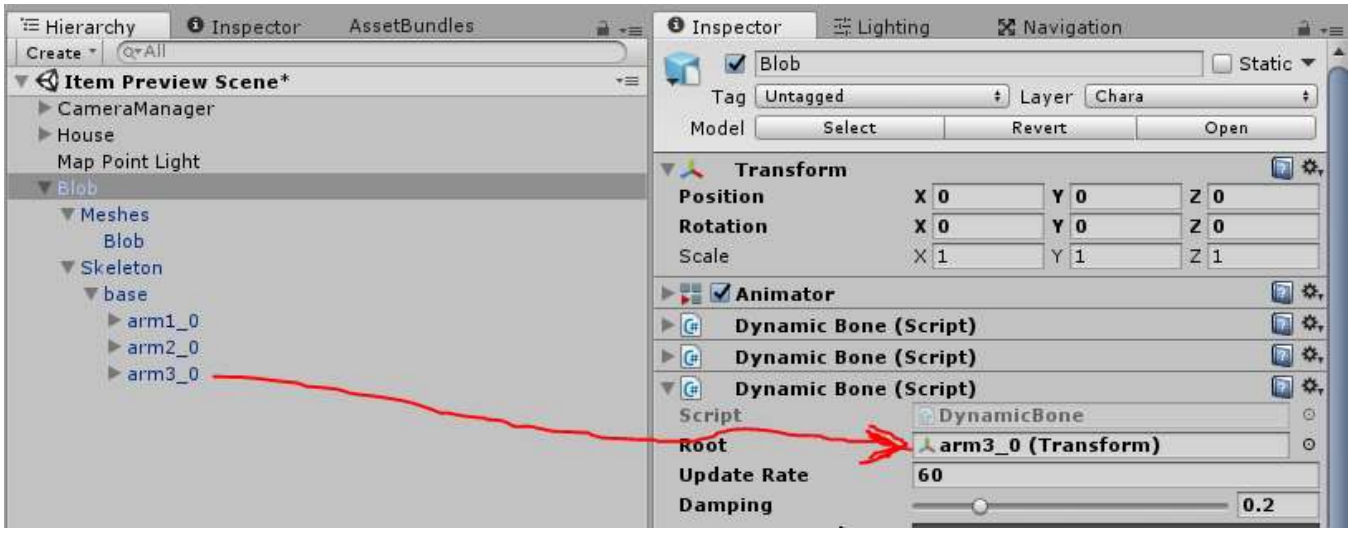
Just like with the unskinned Studio Item, drag your material onto the GameObject with the mesh, then add and fill out the ItemComponent MB. Reminder to fill out the RendNormal array with your Mesh Renderers, and the use color / def color stuff for default colors. I'm going to leave this Blob on the Chara layer, but be sure to change layer if you want your item on the Map layer.

Now for some new stuff: Dynamic Bones. If you aren't familiar, these are components the game uses for things like hair physics and boob jiggle. They go on the root GameObject of your item, same as the MB. You need one Dynamic Bone script for each chain of bones that you want to have the physics, so my Blob needs 3 of these scripts, 1 for each tentacle. How do you know how to fill it out? Well the official documentation is a text document that I don't think is included in KK Modding Tools, but it's posted in the KK Discord under #modder-resource. It tells you what the fields do, but not what size values you should put in there. So to get some ballpark starting values, look at the settings on game items, or the Accessory Hair Example in KK Modding Tools.

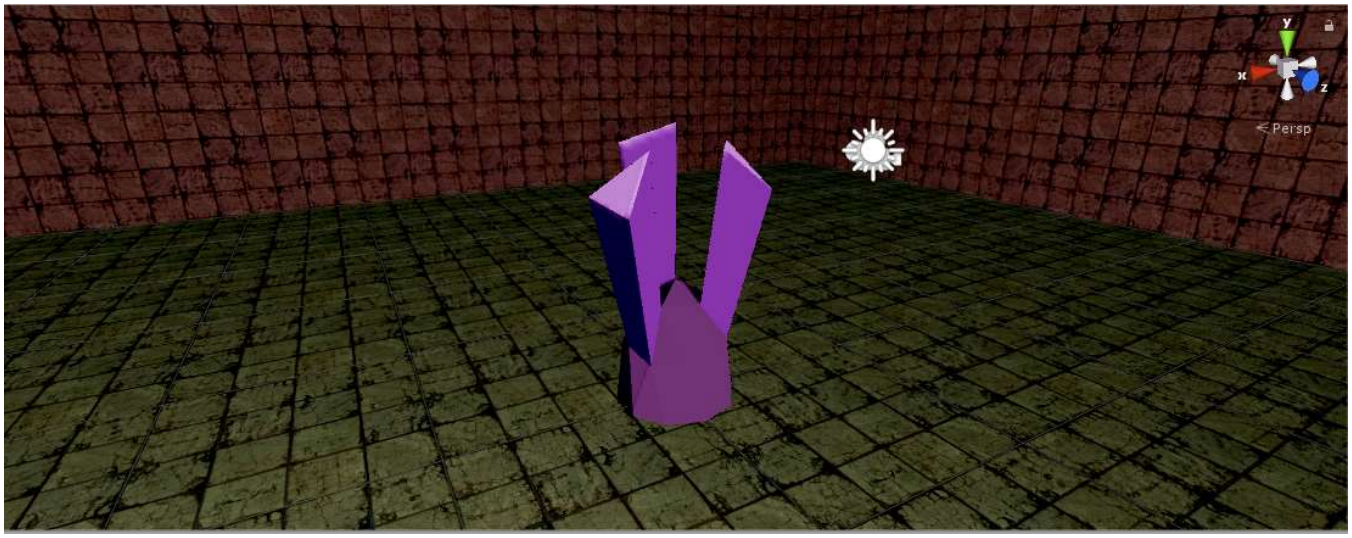
In fact Unity lets you directly copy the component settings from the Accessory Hair Example, and paste them on your own item. So that is what I do.



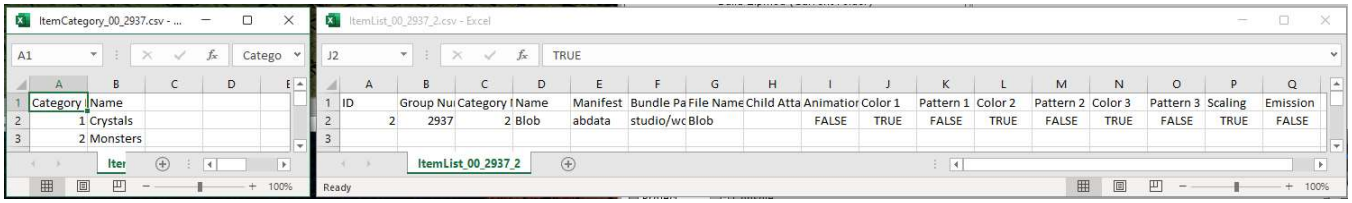
Then I drag in the appropriate GameObjects to the Root Field on each one. I'll leave the settings at that for now and see in Studio if I am happy with the behavior. I don't think you can test them in KK Modding Tools.



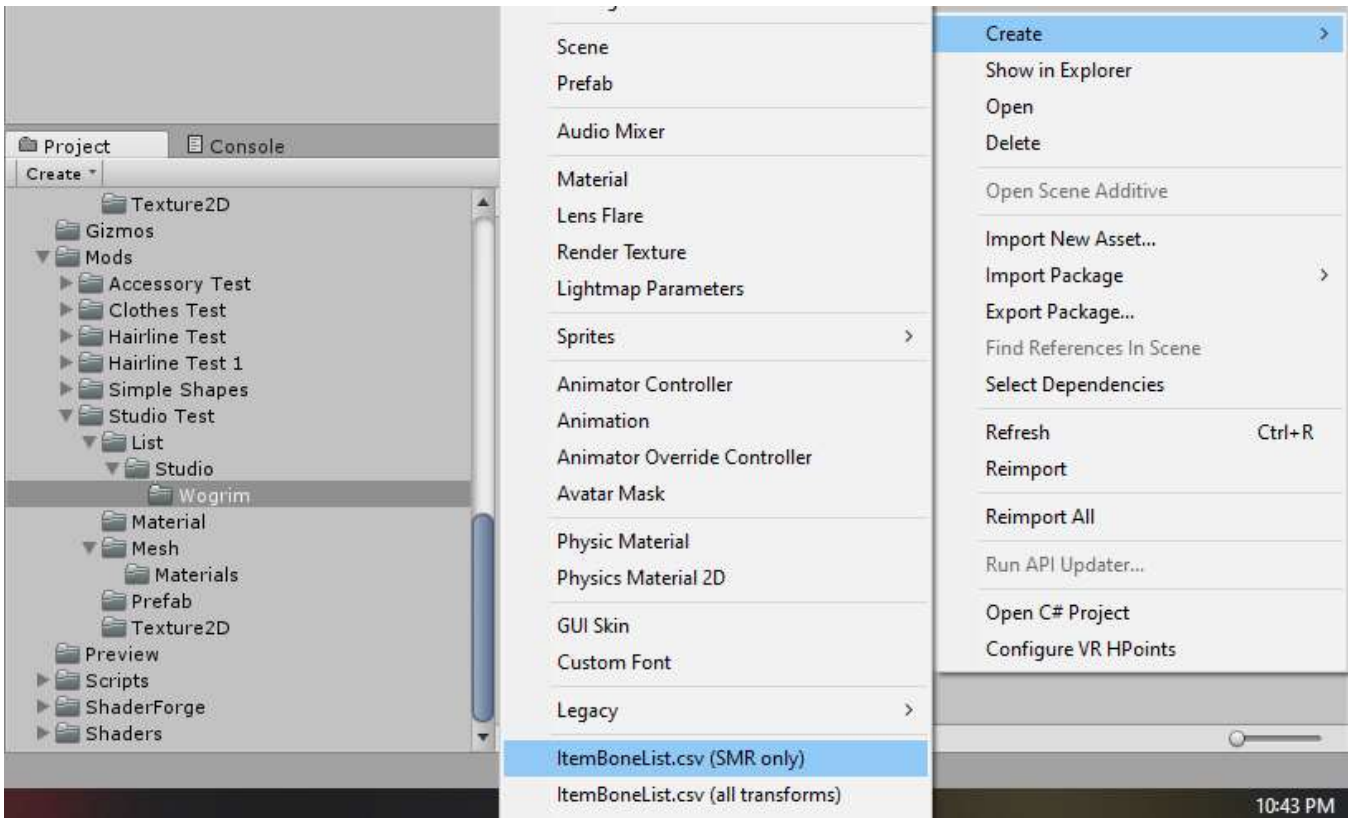
So now I make the prefab, assign it to the same AB as the unskinned item, then bring it back into the scene to give a quick check how it looks with the default colors I set on the MB. I had copied the Crystal material and my low poly blob doesn't look good with the specular and rim lighting, so I change those on the material.



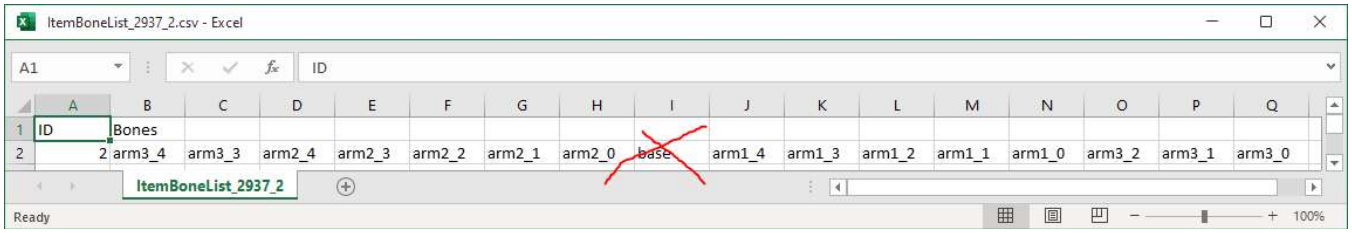
So for list files, I add a category named Monsters, and create a new ItemList file for that category. You can put items in different categories in the same ItemList, but I think a separate ItemList file per category is better organization.



And now for the posable (FK) part of our item to work, we need to do one more thing which is the bone list file. What does it look like? Well I didn't know. But KK Modding Tools can generate it for us. There are 2 options: "all transforms" and "SMR only". Both go through all the items in your list files to create a bone list for each item. "All transforms" just lists all the child GameObjects in your item and adds _generated to the file name; you are supposed to delete the things that are not bones and rename the file. "SMR only" only puts an entry for items with a Skinned Mesh Renderer, and only lists bones that are used by the Skinned Mesh Renderer; it does not add _generated to the name. This is option is not mentioned in the KK Modding Tools wiki, so it may not be reliable and you should check the bone list yourself.



So in the same folder as my other list files, I right click -> Create -> ItemBoneList.csv (SMR only). It creates a bone list file for the crystal's list file, but no item entry because there was no skinned mesh renderer. It creates a bone list file for blob's list file, let's take a look.



It lists all the bones used by the mesh, but I created "base" with the intention that it does not move. People can still effectively move and rotate the base with the controls for the whole studio item, so it is redundant to have FK controls on it, so I remove it from the bone list. Also, I delete the other bone list because there's nothing in it. Now, a quick build of the AB and the zipmod and we're on to testing our item.

Skinned Studio Item: Result

Again, first check that initial size and rotation are as expected, then make sure the item doesn't break when moving/rotating/scaling it. Check simple features before the complicated stuff. I know it is exciting to see the fancier stuff working, but those are a lot less impressive if you forget the basics.

So if you're good to go, your item's dynamic bones will move when you move the item, or if you attach it to something that is animated. Or you can press that FK button in the anim menu and pose your item. So in this screenshot I posed the one by the crystal, and attached another one (which I shrunk down to cute size) to the character's back so the tentacles wave back and forth with the character's run animation.