

Mesh Extractor

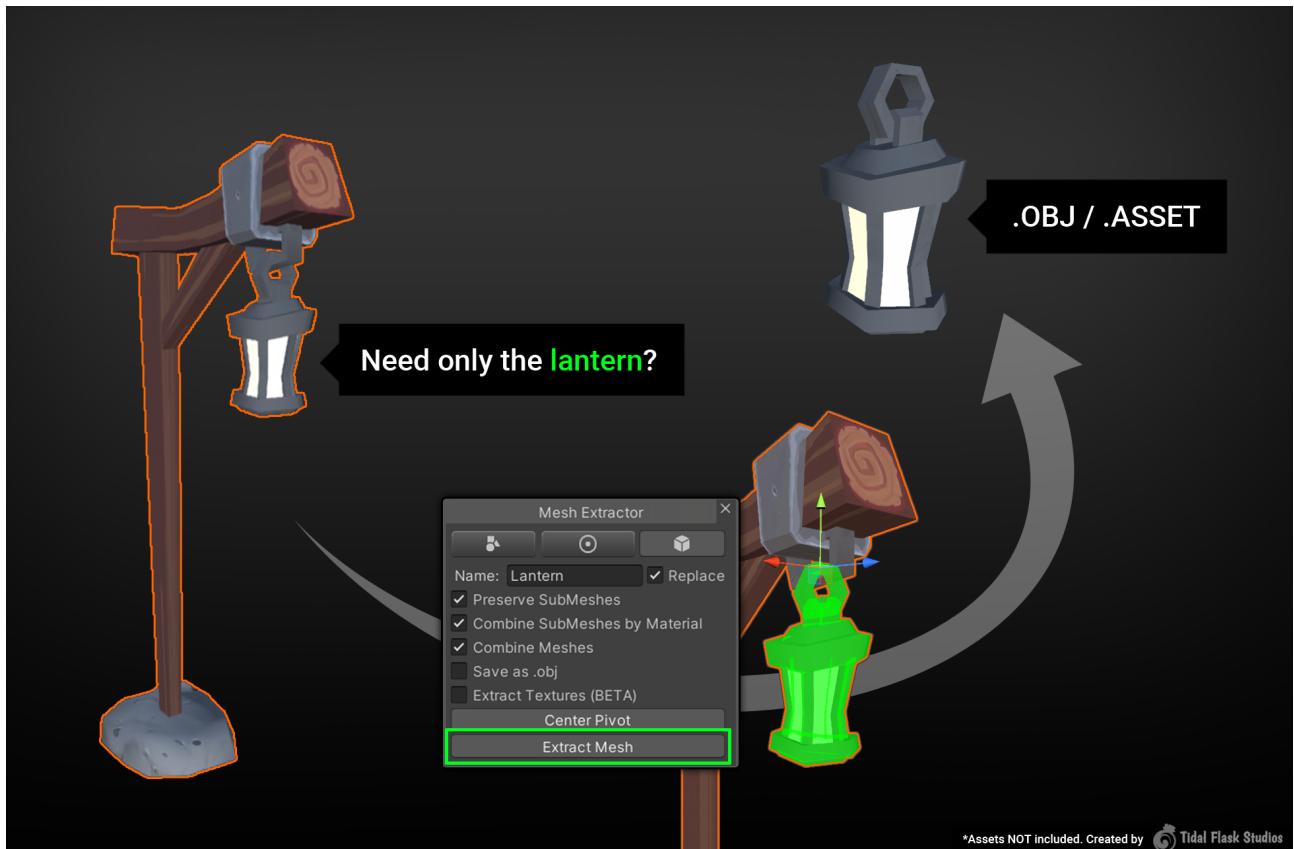


Table of contents

What's it good for?	2
Usage	3
Start the tool.....	3
Select objects.....	3
Select triangles.....	4
Extract Mesh.....	8
About extracting textures	11
Bones / Rigs / Armatures	12
Exporting meshes with bones.....	12
Things you should know about bones in Unity (aka „Why we need that BoneData asset.”).....	13
Frequently Asked Questions (FAQ)	16
How to mix skinned meshes?.....	16

What's it good for?

If you are like me then over time you have purchased a lot of awesome assets on the AssetStore. But often you only need a small part of it.

Let's take this asset from [TidalFlask](#) for example. It's a great looking asset but what I need is just one sheet and a nail. Sadly it's all delivered in one single mesh.



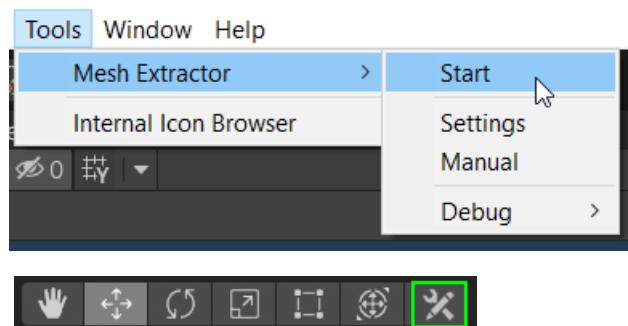
With Mesh Extractor I can get the what I need within one minute. No modelling software is needed.



Usage

Start the tool

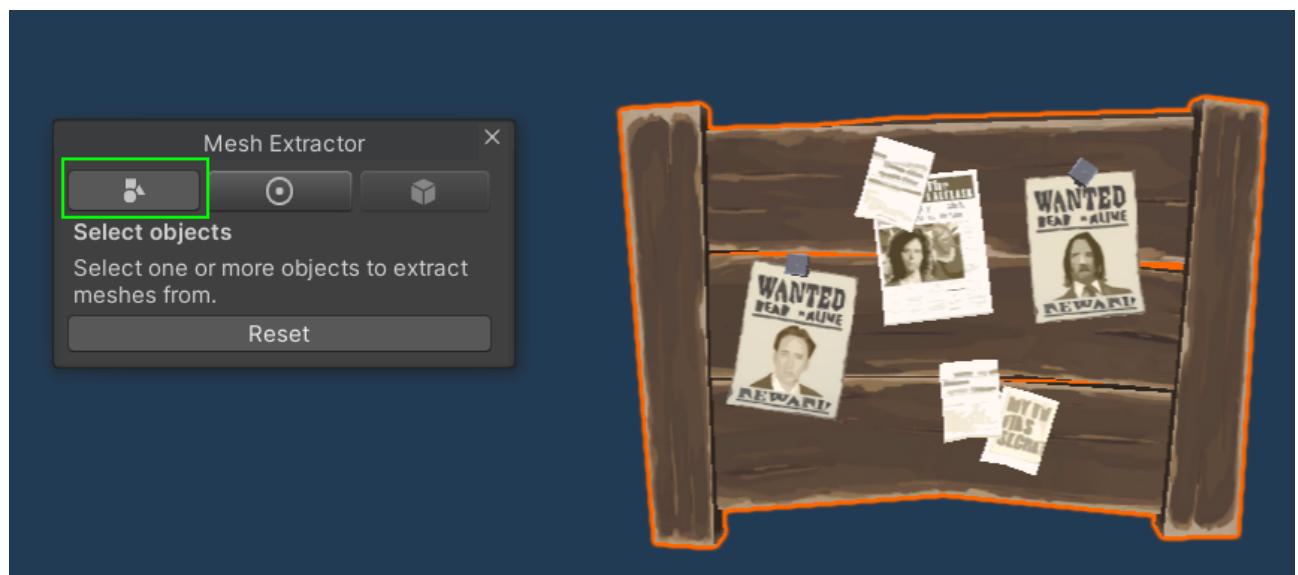
Open the tool via **Tools > Mesh Extractor > Start** (or via the Tools bar).



Select objects

Usually the tool starts in the „Select Objects“ mode.

Here you should select one or more objects to extract from. We do this so we don't accidentally select parts of other meshes behind the object.

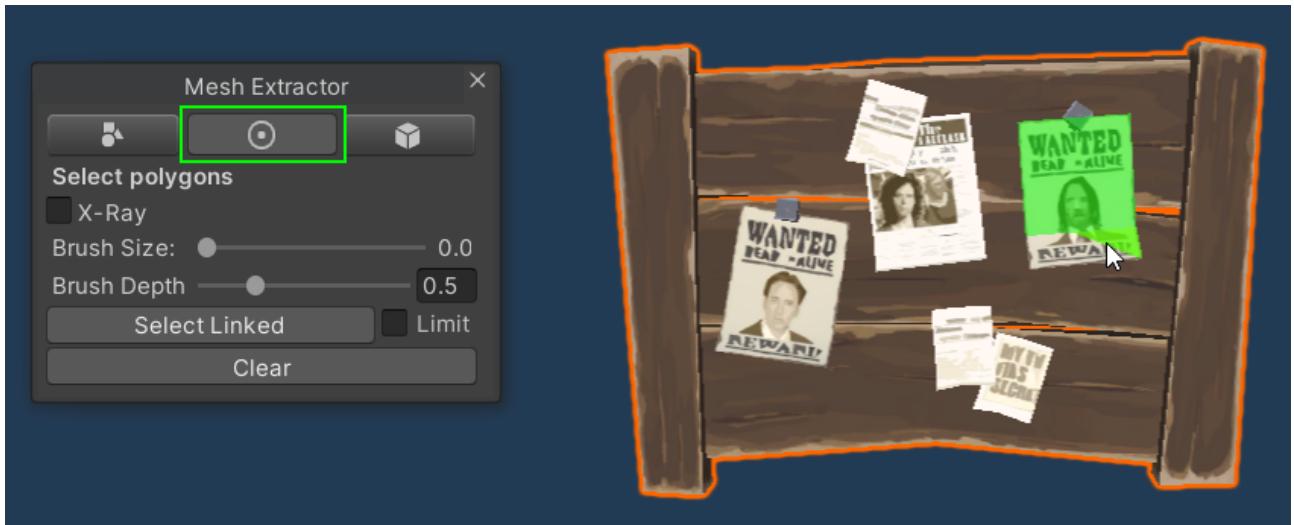


Reset: Clears the current selection, deselects any object and resets all configurations to default.

Select triangles

Once the object is selected you can start selecting the triangles which you want to extract. Simply click or drag your mouse over the object. Press **CTRL** to erase your selection.

 **HINT:** You can add new objects to extract from while painting if you hold **CTRL** and click an object. Though to remove it you will have to switch to the select objects mode.

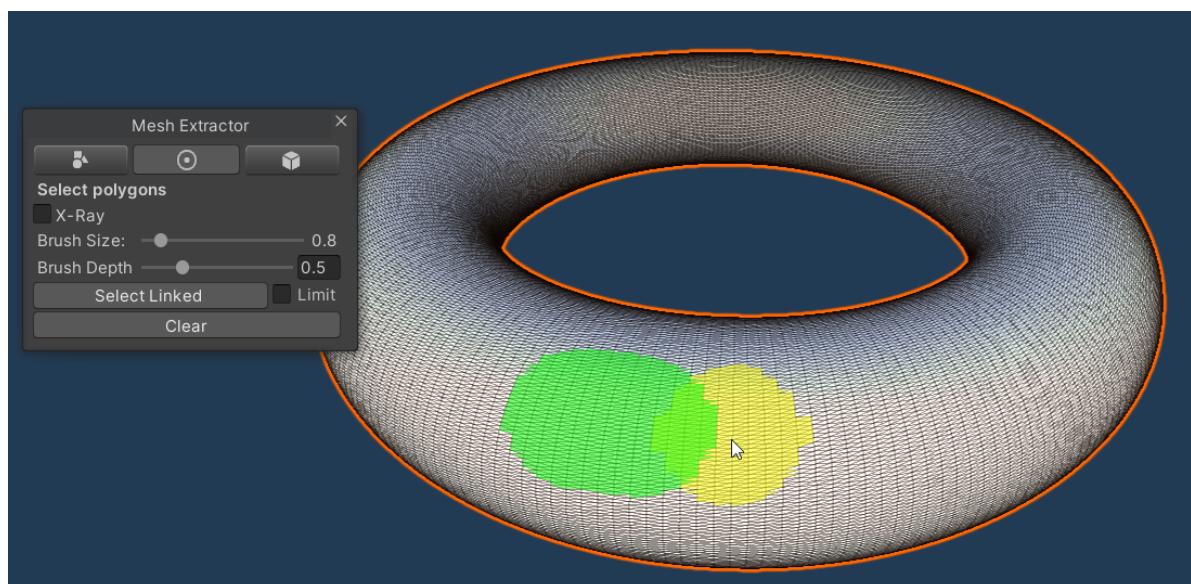


X-Ray: X-Ray mode allows you to select front and back facing triangles at the same time.

Brush Size: Reduce the brush size to 0 to select only one triangle at a time. You can also use **SHIFT + MOUSE WHEEL** to change the brush size.

Personally I mostly use brush size 0 in combination with the „Select Linked“ button (see below).

Brush sizes greater than zero are useful for high poly meshes where selecting single triangles would be too cumbersome. Like in this scenario:



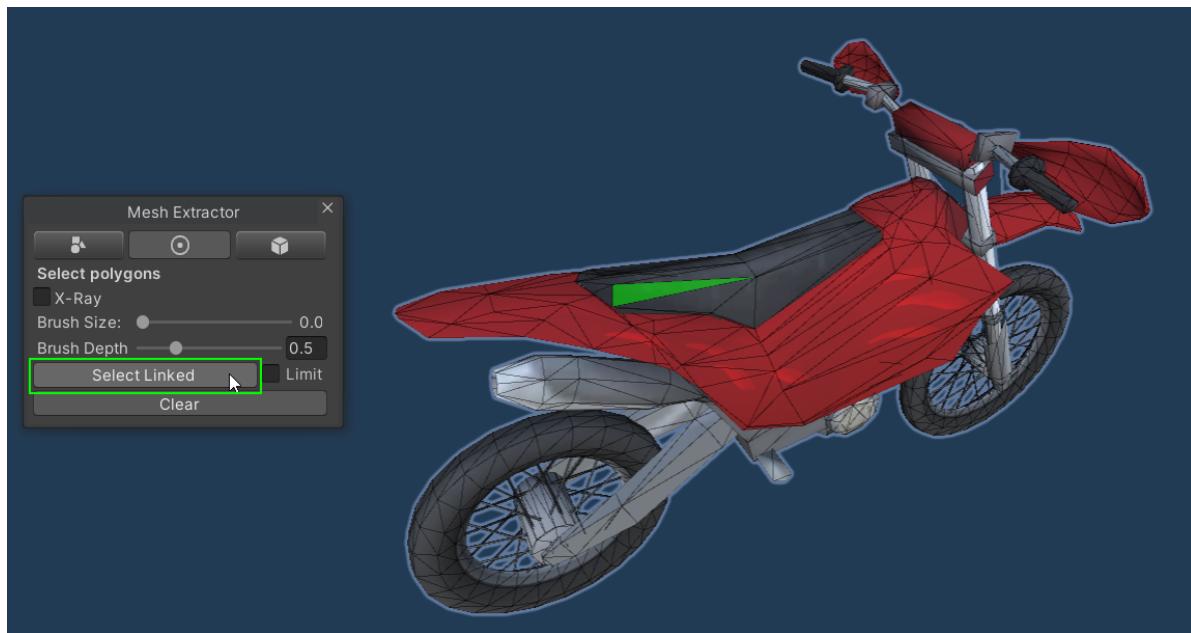
Brush Depth: Brush depth defines how far into the object the selection will go. This helps to avoid selecting background polygons by accident. If you want infinite depth then simply turn on X-Ray.

Select Linked: Selects all triangles which are connected to the last selected triangle.

This one may need some more explanation. In many cases selecting single triangles is a lot of work and actually what you want to select is a part of a mesh which has triangles sharing the same vertices (meaning one triangle is connected to another triangle with at least one shared corner point).

Example: Here we want to select only the seat of the bike.

We select only one triangle of the seat and then hit the „Select Linked“ button.



Et voilá, the tool analyzed the mesh and selected only the part we wanted. but HOW?

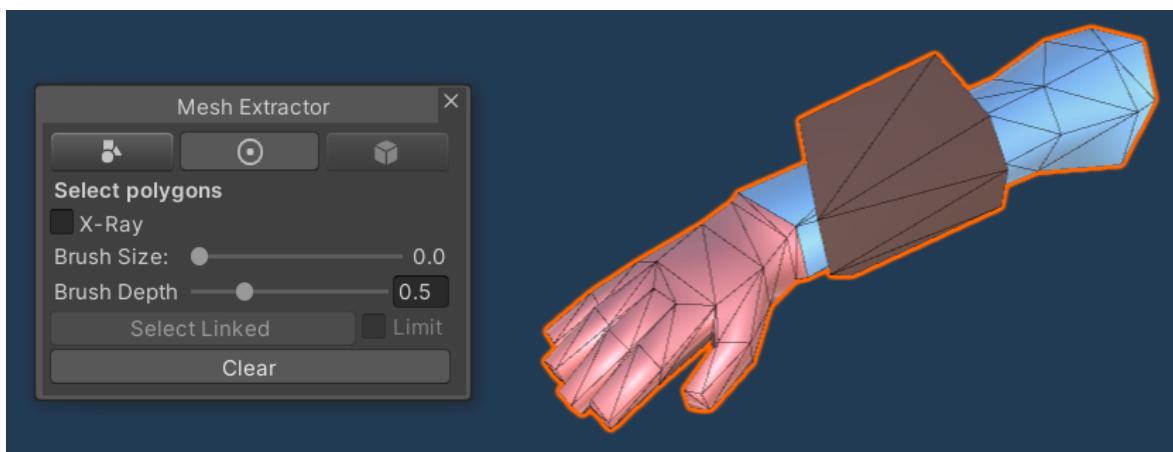


If we look closely we see that the seat mesh actually does NOT connect with the bike body. The tool can use that information.

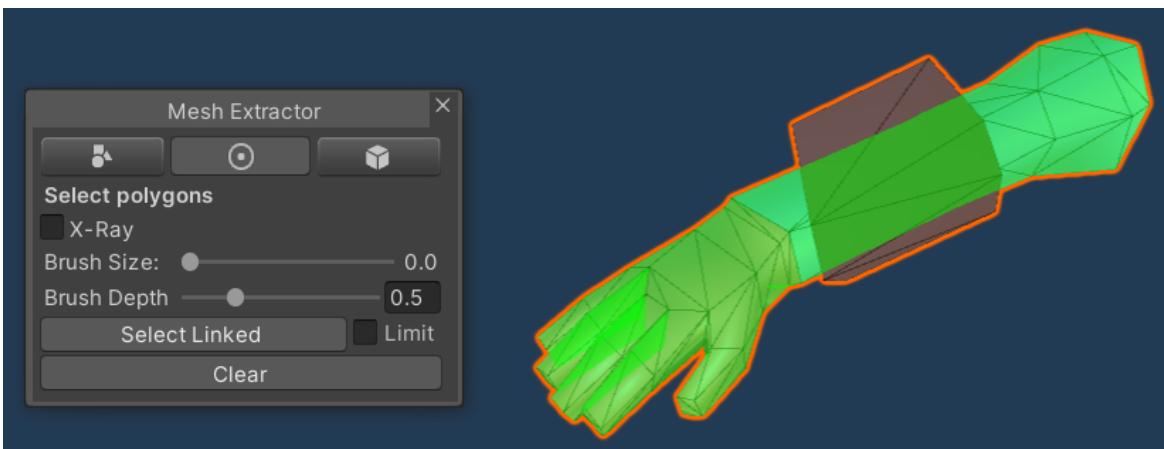


Select Linked > Limit: Enable this to limit the selection to a single sub mesh. It will use the sub mesh of the last selected triangle.

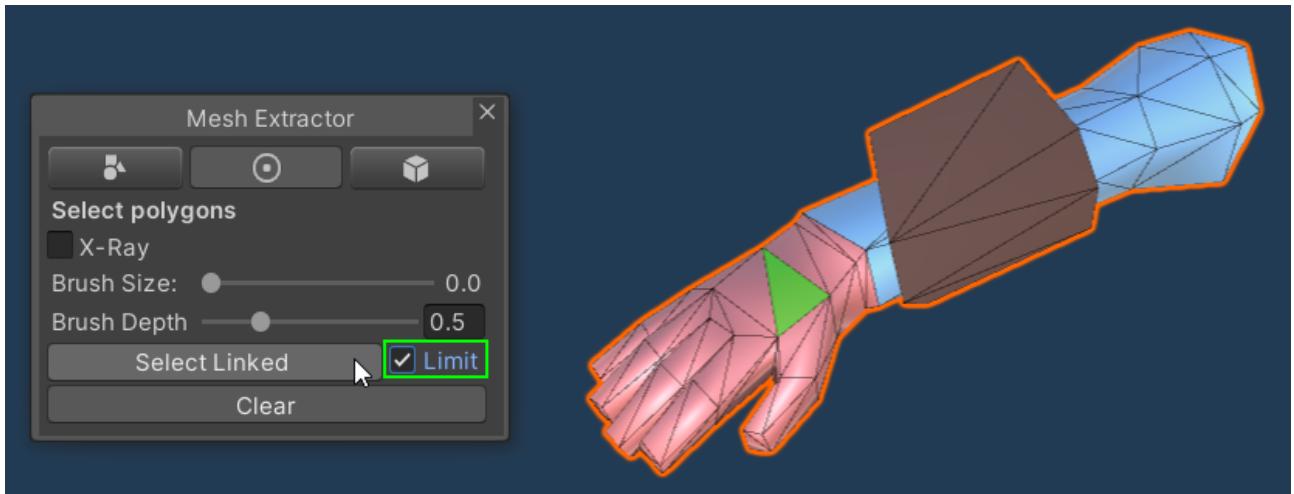
Again this one needs some more explanation. Let's take this mesh for example.



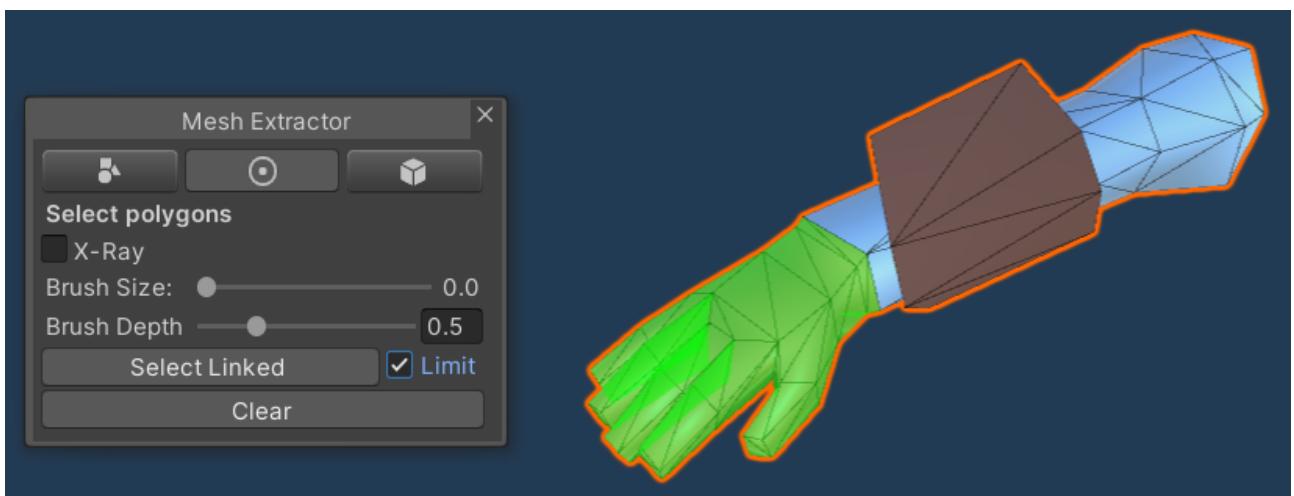
All the arms vertices are connected. If we use „Select Linked“ on it the whole arm will be selected. Like this:



We can see from the assigned materials that the arm has some sub meshes (one for the arm, one for the hand). Let's enable **Limit** to limit the connected selection to a sub mesh.



This is the result if the „Select Linked“ options is limited to a sub mesh.



Clear: Clears the current selection.

Extract Mesh

After selecting the triangles we are ready to extract the mesh.

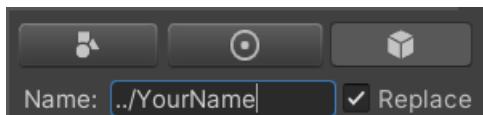


Adjust the Pivot: You will now see a move gizmo. This is the position of the pivot. You can move it to wherever you want.

HINT: You can hold down the V key to snap the pivot to a vertex.

Name: The base name of the newly generated assets. The assets are saved under **Assets/ExtractedMeshes/[TheNameYouEnter]**. You can change the base path „ExtractedMeshes“ in the settings (Tools > MeshExtractor > Settings : Extracted Files Location).

HINT: You can use relative paths here too. These are then relative to the „Extracted Files Location“. So if you want your assets to be stored directly in the Assets/ folder then simply prepend a „..“/. Like this:



Replace: If enabled then the new prefab will replace the old. If disabled the new prefab will be stored with a new name.

Preserve SubMeshes: Enable to preserve sub meshes in the new mesh. If disabled then all sub meshes within one renderer will be merged into a single mesh.

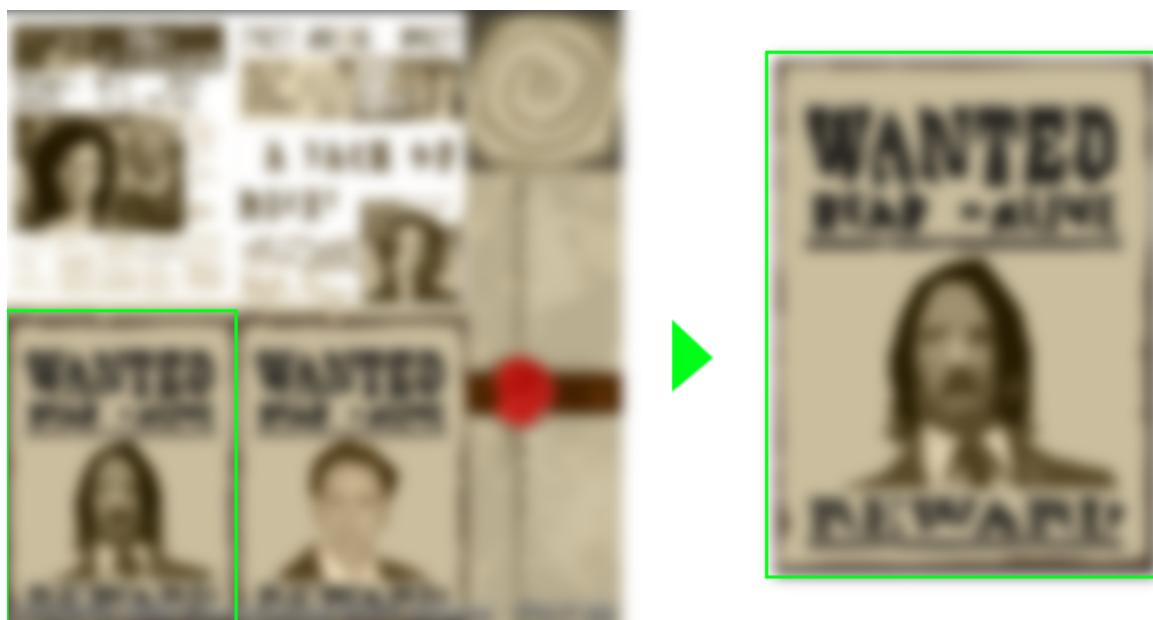
Combine SubMeshes by Material: If multiple sub meshes have the same material assigned to them then these will be merged into one submesh if this option is enabled. This has no effect if 'Preserve SubMeshes' is disabled.

Save as .obj: Export the mesh as .obj & .mtl files instead of an .asset file.
NOTICE: The obj format does only support one set of Uvs and no extra info like bone weights. If you have „Extract Bone Weights“ enabled then the „Save as obj“ option will not be available. Uncheck it to save as .obj.

Extract Texture: Extract the parts of the texture which are used by the selection and creates a new (possibly smaller) texture from it.

NOTICE: This feature is experimental. The reduction of the texture size depends on the original UV layout (it uses a bounding box).

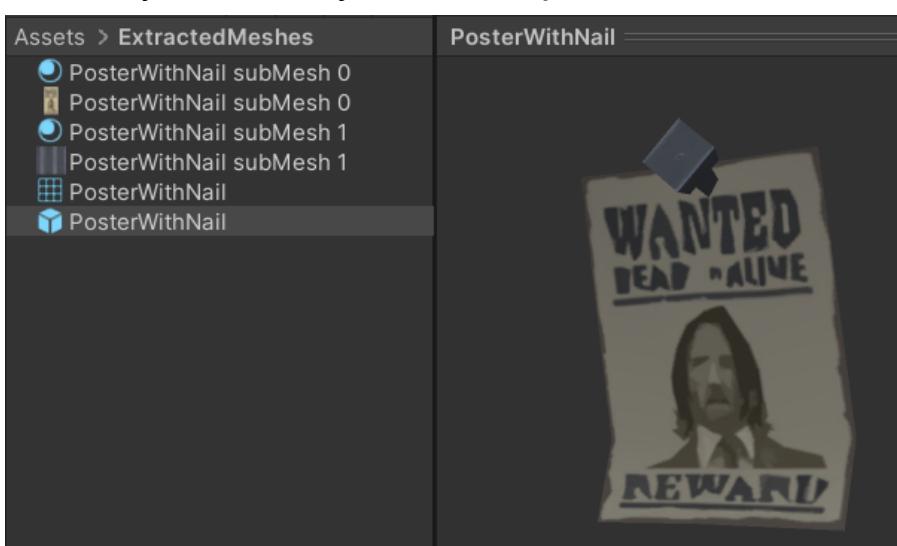
„Extract Texture“ reduces the size of the texture by copying only the parts that are needed.



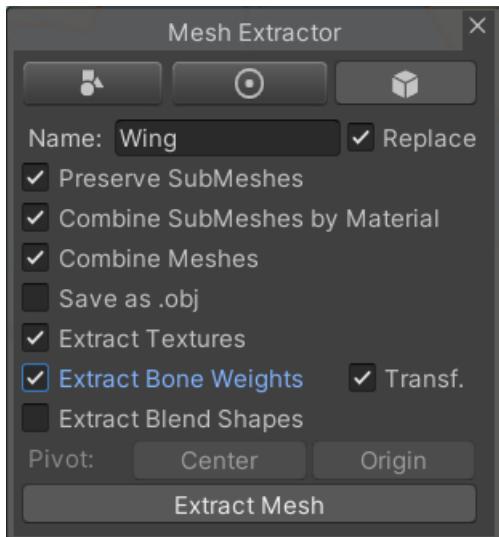
(The image is blurry on purpose to not reveal a third party publishers texture).

Center Pivot: Recenters the pivot in the middle of all selected triangles.

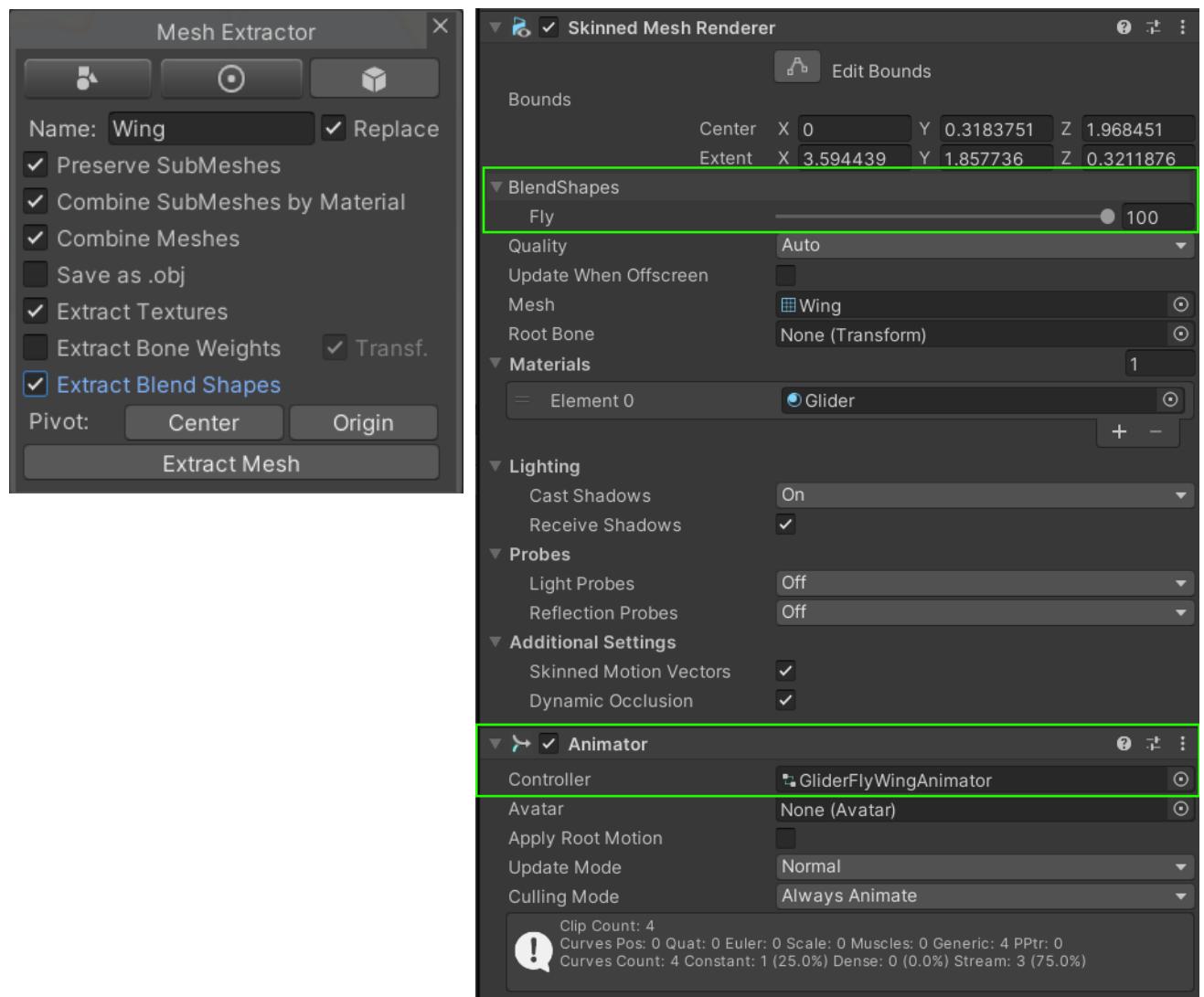
And finally this is what you will end up with:



Extract Bone Weights: Extract bone weights (more on that below).



Extract Blend Shapes: Extract blend shapes. If your renderer has an Animator component attached then it will also copy that.



About extracting textures

Textures are connected to 3D models through materials. Each material uses a shader and within that shader are slots (properties) which can take a texture. These slots are named. A common name used by most Unity Standard shaders is „_BaseMap” for the main albedo texture.

Since every shader developer can pick these names freely it is simply impossible to guess all the names. For example one developer may call the albedo texture „_MainTex” and another simply „_Albedo”. The Mesh Extractor searches for some of the most common names used by Unity devs. It should work fine with most Unity Standard shaders (though even Unity sometimes mixes up the names). It will probably fail to find textures on custom shaders.

Here is the list of searched shader property names for each texture type:

Albedo (main texture): „_BaseMap”, „_MainTex”, „_AlbedoMap”, „_AlbedoTex”, „_Main”, „_Albedo”

Normal Map: „_BumpMap”, „_NormalMap”, „_Bump”, „_Normal”, „_MainNormalMap”, „_ParallaxMap”

Specular Map: „_SpecGlossMap”, „_SpecularColorMap”, „_SpecularMap”, „_Specular”, „_MainSpecularMap”

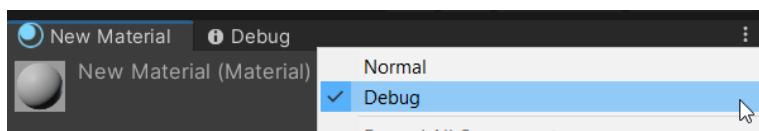
Metallic Map: „_MetallicGlossMap”, „_MetallicColorMap”, „_MetallicMap”, „_Metallic”, „_MainMetallicMap”

Emission Map: „_EmissionMap”, „_EmissiveColorMap”, „_Emission”, „_EmissiveMap”, „_Emissive”, „_MainEmissiveMap”

Occlusion Map: „_OcclusionMap”, „_OcclusionColorMap”, „_Occlusion”, „_MainOcclusionMap”

You can extend this list by editing the „Assets\Kamgam\MeshExtractor\Editor\MaterialPropertyExtensions.cs” file.

If you are not sure how to find out which property names your custom shader is using then make a new material and assign the shader to it. Then click on the material and in the inspector change the mode to „Debug”.



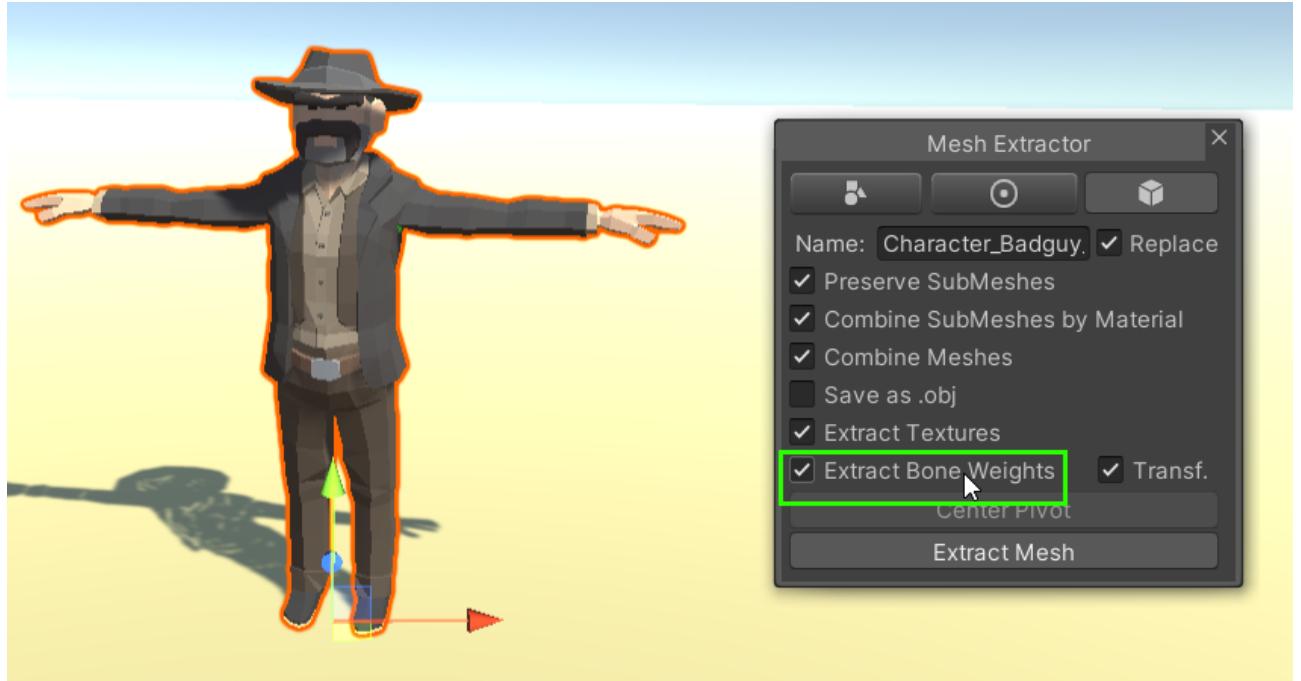
If you scroll down to the „Saved Properties” List you will find the property names listed right there.



Bones / Rigs / Armatures

Exporting meshes with bones

To export bone weights tick the „Extract Bone Weights“ box.



By default the „Transforms“ export is also checked. It exports the bones and puts them into the prefab.

However if you wish to only export the mesh and boneData then you can untick it. Just be aware that it will set the bone transforms to null to avoid 'Bones do not match bindpose' errors. You will have to link the renderer to new bones manually afterwards (or use the BoneDataResolver). More on that in the „Things you should know about bones in Unity“ section.

Things you should know about bones in Unity (aka „Why we need that BoneData asset.“)

One important thing to note about bone information in Unity is that it is NOT stored in the mesh but in multiple locations. Bones require three things to work:

1) **Bone weights** for each vertex (stored in the mesh). They tell each vertex how much influence a bone has on it.

2) **Transforms** to control the movement of the vertices (stored as objects in the hierarchy). That's the thing you will probably think of if you hear the word „bone“.

3) **Something to connect the bone weights to the transforms.**

Now here it becomes tricky since that „connection“ information is kinda divided. It is stored as a hidden list in the SkinnedMeshRenderer. - You look confused. Let me explain.

To constitute a bone you need two things (A and B):

3.A) First are the bind poses which are stored as a list (array) in the mesh.

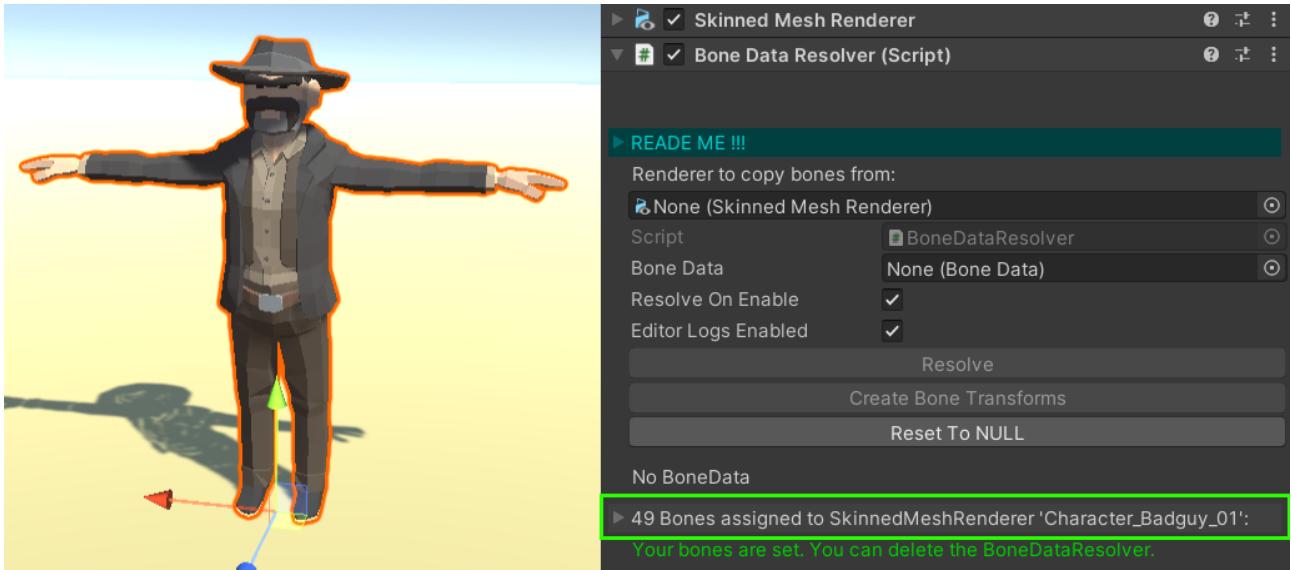
Think of each bind pose as a thing (matrix) which connects multiple vertices to a transform (bone). It is responsible for transforming the vertex position depending on the bone position, rotation and scale. It's NOT the bone transform you see in the hierarchy, rather it's like a thread that's connecting the vertices to whichever positional data it is given.

3.B) The thing that most commonly is referred to as a „bone“ is just a transform (position, rotation and scale information). In Unity this resides in the hierarchy within the scene file (that's what you move, rotate and scale). If a transform is changed then the bind pose will take notice and it then moves the vertices accordingly.

As mentioned before these bone transforms are NOT saved in the mesh. They are in the scene. How are they connected to the mesh?

Well, the SkinnedMeshRenderer has an internal list which tells it what scene transforms match to what bind poses. For some reason Unity decided to hide this list from the inspector. Sadly you do not even see it in the debug mode.

To deal with the hidden „transform → bind pose“ information I've created the „BoneDataResolver“ component which makes it visible in the inspector:

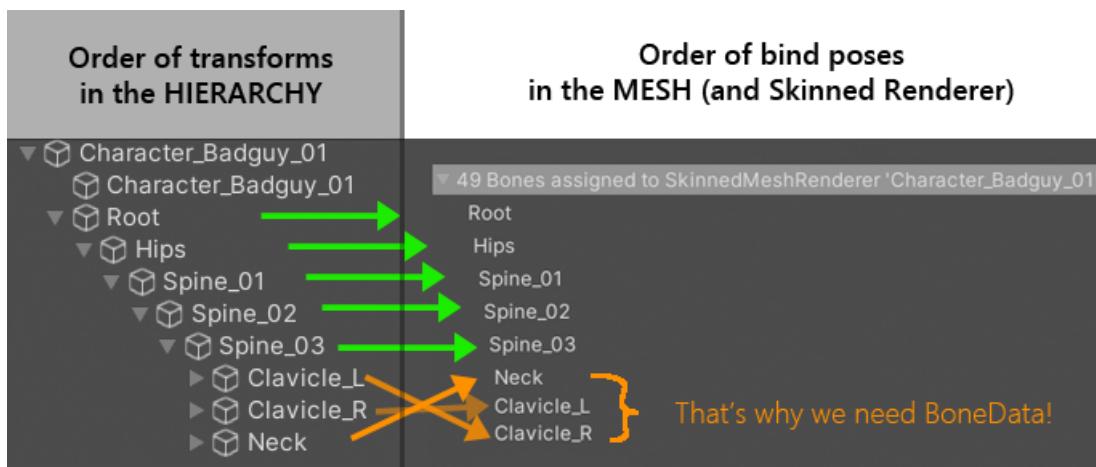


Unfolding the „... Bones assigned to ..“ section reveals the connections:



Now here is why the paragraphs above are talking about the ORDER being important. Often the order of the bone transforms in the hierarchy does NOT match the order in which the bind poses have been defined in the mesh.

Like in this example:



Simply put: the left side of the image above is not enough information to restore the right side.

Now that is a problem. If we export only the mesh we can not restore the **order** information. Even if we have a copy of the hierarchy transforms in the scene we can not do it. It's hidden inside the SkinnedMeshRenderer.

Aaand that's why the **BoneData** asset exists. It stores the ORDER, NAME and HIERARCHY of the bone transforms. Like hits:

Script	BoneData
Root Name	Root
Paths	49
= Element 0	
= Element 1	Hips
= Element 2	Hips/Spine_01
= Element 3	Hips/Spine_01/Spine_02
= Element 4	Hips/Spine_01/Spine_02/Spine_03
= Element 5	Hips/Spine_01/Spine_02/Spine_03/Neck
= Element 6	Hips/Spine_01/Spine_02/Spine_03/Neck/Head
= Element 7	Hips/Spine_01/Spine_02/Spine_03/Neck/Head
= Element 8	Hips/Spine_01/Spine_02/Spine_03/Neck/Head
= Element 9	Hips/Spine_01/Spine_02/Spine_03/Clavicle_L
= Element 10	Hips/Spine_01/Spine_02/Spine_03/Clavicle_L/
= Element 11	Hips/Spine_01/Spine_02/Spine_03/Clavicle_L/

With the BoneData we actually can restore the order information.

Btw.: That's what the „Resolve“ button on the BoneDataResolver does for you. It searches in the hierarchy for the „Root“ transform and if it finds one it uses that to recreate that hidden list in the renderer (effectively connecting the transforms to the mesh vertices).

Once you have restore that hidden list you can delete the BoneDataResolver (unless you plan to change the list again, then I would keep it around).

Phew, thanks for reading it all.

I hope that was understandable.

Frequently Asked Questions (FAQ)

How to mix skinned meshes?

In this example video we will mix two skinned meshes from one of the [Synty asset packs](#). You can find a video of how to do it here: <https://youtu.be/N6zGRhmtKNI>

