Intro to Unity for Modders

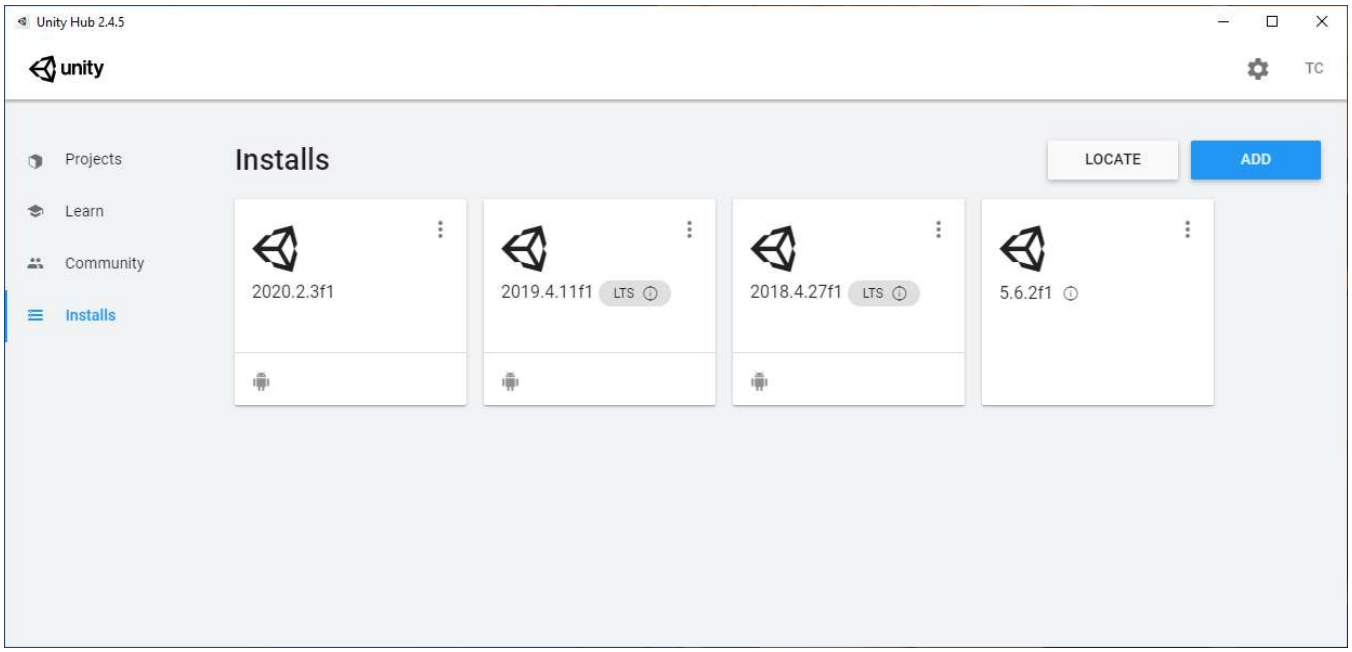
Wogrim's Brief Guide to Unity for Modders, with KK Modding Tools

What is this Guide About?

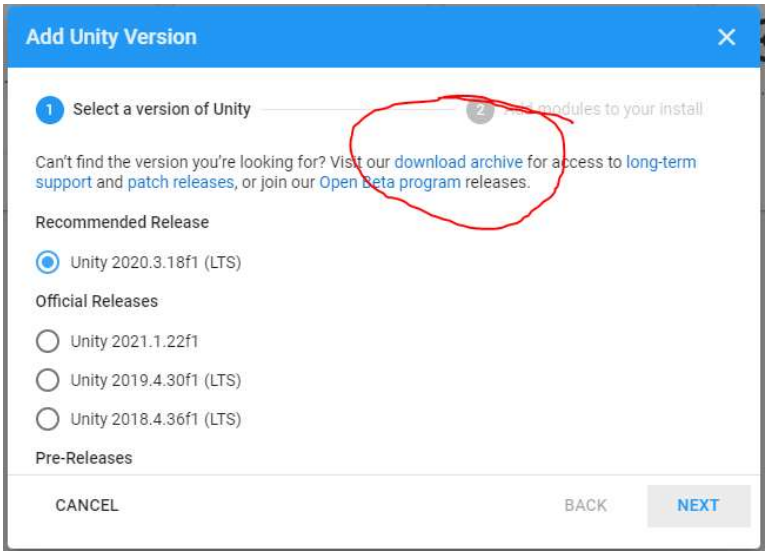
This guide is to cover basics of installing and using Unity if you've never touched it before (just the stuff that matters for making the most common types of mods). It will also talk about how to install and use KK Modding Tools. You should probably have a basic understanding of a Zipmod before you read this guide, but you could probably make it through the install and interface sections with no knowledge.

Installing Unity and KK Modding Tools

So if you go out and try to download Unity, you'll probably get something called Unity Hub, which is a launcher for the Unity Editors. Unity is the game engine, you make a game (or in our case mods) with a Unity Editor. But there's a bunch of different versions, and you may have projects (games or other things you're creating) in different versions, so they created a launcher called Unity Hub to manage it all. So find that and install it, it's pretty small unless it comes with a Unity Editor, I don't remember. IIRC you have to make an account and log in, and you use the free version unless you make over $100k per year from it.

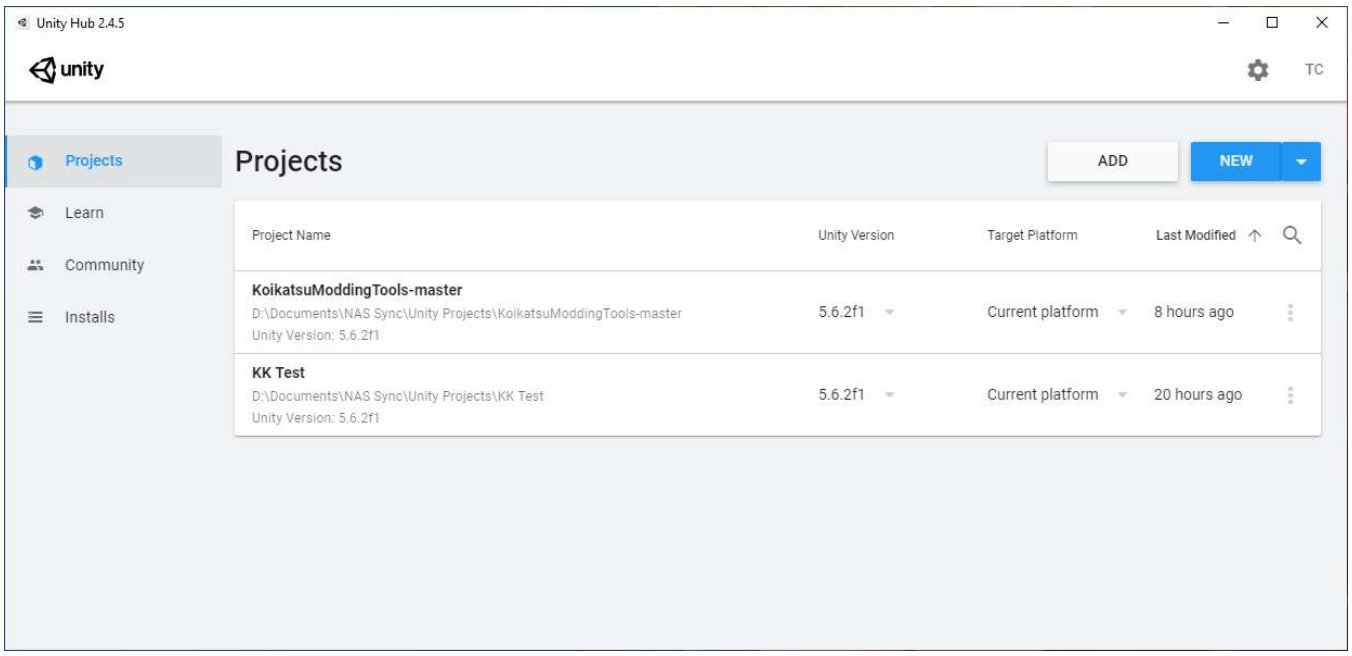


So you won't have any projects yet, and you won't have any Unity Editors installed. Go to the **Installs** tab and click **ADD** and a window will pop up asking which version you want to install. For making ABs for KK mods (and for using the KK Modding Tools project) you must get version **5.6.2.f1** which won't be in the list. So hit the link for the **download archive** and find it on that page (it just says 5.6.2) to get it.

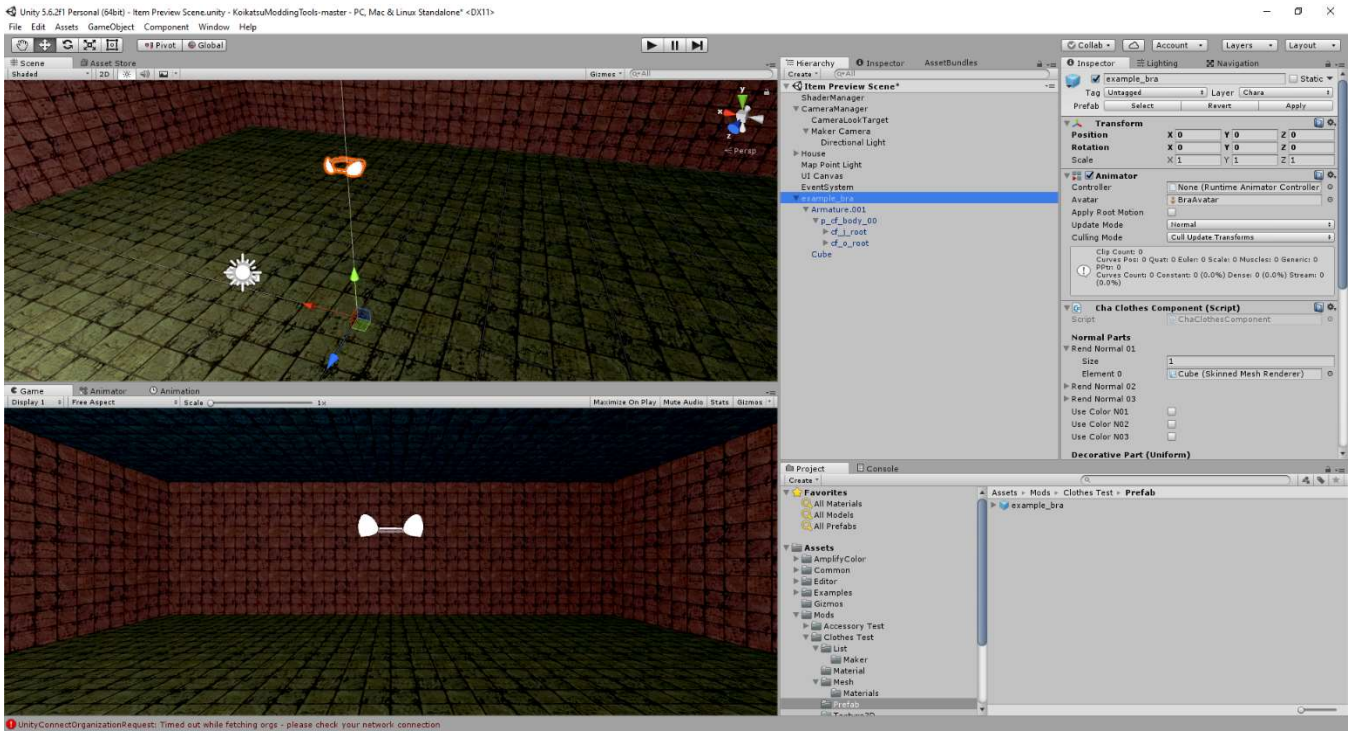


So at this point you should have it installed, and you could create a project and make mods with it, but KK Modding Tools makes most types of mods significantly easier/faster so download the zip for it from the github: https://github.com/IllusionMods/KoikatsuModdingTools

Extract the zip somewhere, and on the **Projects** tab in Unity Hub, click **ADD** and navigate to that extracted KK Modding Tools folder and **Select Folder** on it. Then the project should show up in Unity Hub and you can open it.



Unity Interface Stuff



So when you've opened KK Modding Tools it should look something like this but less stuff. There's a bunch of docked windows, and if you need another one, find it under **Window** on the menu at the top. So, for an explanation of the important windows:

Project contains the folder structure for your project. Well, specifically the **Assets** folder which is where you add things when making a game (or mods). Some things you can create with a right-click in the Project window, other things you import with a drag-and-drop from Explorer. The project contains other stuff outside the Assets folder but you shouldn't mess with it. KK Modding Tools comes with a few folders under Assets, but for making mods you basically only make changes to the Mods folder.

Scene shows the currently loaded scene (usually a game level) in an editing-friendly way. The first time you open KK Modding Tools there will probably be no scene loaded, and you should find "Item Preview Scene" in the Assets folder. By default it will be pretty empty, but you can add stuff to it if you want. Pressing F will focus the camera on the currently selected object, and holding Alt + left mouse button drag will rotate the camera around it.

Hierarchy shows the structure of things in the scene. IIRC the Item Preview Scene only has a camera and a light, with the light being a "child" of the camera (the camera is the "parent" of the light). This means whenever you move the camera, the light moves with it. Everything listed in the Hierarchy window is a **Game Object** whether it's something physical or not, and every Game Object has a **Transform**, which is the combination of the item's location, rotation, and scale. The Transform is irrelevant to some Game Objects, but every Game Object has one. The Transform is a **Component** of the Game Object, but you can add other Components (such as MBs) to make Game Objects do things.

Game gives the gameplay camera view of the current scene, which is great for playtesting a game, but mostly useless for making mods. I'm only mentioning it because I think it's one of the default windows.

Inspector is where you edit the Components of the currently selected Game Object or most things in the Assets folder, such as **Import Settings** for textures and meshes.

AssetBundles is a special window created by KK Modding Tools for building ABs and mods. Fill in your game's install location so it can copy the mod to your game's mods folder when built.

Console will show error messages, and also shows messages when KK Modding Tools creates (or tries to create) ABs and mods. Check it if things aren't working.

Texture Item Zipmod Process with KK Modding Tools

Note: This is just the process in KK Modding Tools for someone who already knows what this type of mod entails.

In the Project window, duplicate (Ctrl+D) the "Body Paint Example" folder in Assets/Examples (because it is a texture item). Move the new folder into Assets/Mods and rename it to whatever you're going to call your mod. A single click on the manifest in this folder will give you a special editor in the Inspector window which makes filling out the manifest a little easier.

Delete existing textures in the Texture2D folder and the Thumbnail folder, and then import your own textures. Adjust import settings if needed. For the first texture, in the bottom of the Inspector window assign a New AB called **chara/author/modname.unity3d** (so if I made a nose mod, it might be chara/wogrim/noses.unity3d). Assign the rest of your textures to this AB (or put thumbnails in a different AB, typically under **chara/thumb/author/modname.unity3d**). These AB names will automatically convert to lowercase.

The list file must remain in List/Maker for KK Modding Tools to find it and place in the correct location of the zipmod, but rename it to whatever item type your mod is. Then right click -> Show in Explorer and open it from there instead of from within Unity, or else Unity might try to open it with some other program. Change the list file to the proper format for your item type and fill it out, with the AB names that you assigned your textures to and the name in Unity those textures have (no file extension).

Go to the AssetBundles window, and fill in your game install location if you haven't already. Then hit the Build Asset Bundles button (which will take a minute first time because of ABs for the Examples) and you should get a message in the Console that ABs are built. Then with the folder for your mod selected in the Project window, click the Build Zipmod (Current Folder) button in the AssetBundles window and it should tell you the mod was built. Assuming you have the Copy Mods checkbox marked, you should now be able to start up your game and see the item.

You don't have a mod folder quite the same as not using KK Modding Tools so troubleshooting may seem harder, but if it built the zipmod you can pull it out of your game's mods folder and extract it to see what happened (such as it won't have a list file if you renamed its parent folder, or the AB will be messed up / missing if you didn't assign stuff properly).

3D Item Zipmod Process with KK Modding Tools

Note: This is just the process in KK Modding Tools for someone who already knows what this type of mod entails. I'm assuming you've read the previous section so I won't re-explain how to do things that are the same.

For 3D items (Accessory, Hair, Clothes, Studio Item) duplicate appropriate example folder and move to Mods (and rename). Delete everything but folders, manifest, and list file(s). Fill out the manifest.

Import your item's FBX. You may need to change Import Settings such as File Scale. If a material comes with the import, textures may have come with it, otherwise you need to manually import them separately. If there was no material, create a new one in your mod's Material folder (Project window). In the Inspector window, change the material to the appropriate (placeholder) shader for your item and fill in the textures and other settings.

Drag the imported mesh from the Project window into the Hierarchy window. If your item has size or rotation problems, you may be able to fix it later or you may not. It's also possible that you think you have a problem but don't. So keep reading, but you may have to go back and redo the process with different import (Unity) or export (modeling software) settings.

Any bones the item had are child Game Objects, and wherever there was a mesh there is now a Game Object with a Mesh Renderer Component. KK Modding Tools automatically puts items (with all children) in the Chara layer. If you're doing a Studio Item, you may want to change it (with all children) to the Map layer (upper right in Inspector window).

Drag your material from the Project window onto the Game Object with the Mesh Renderer in the Hierarchy window to apply it. One of the great features of KK Modding Tools is there is a script that automatically swaps the placeholder shader to the real shader ("preview" shader) so you can get a good idea how your item will look in-game. But there are some things to note.

- there are technical limitations that require us to use the placeholder shader
- if you save scene when something has the preview shader on it, it will revert to the placeholder shader to not break your item, there's a couple ways to get it to switch back to preview shader like deleting the item from the scene and re-drag it in there
- do not do File -> Save Project with an item in the scene, it breaks the shader on the item's material and you have to manually change it back

You will later be dragging a Game Object from the Hierarchy window into your mod's Prefabs folder (in the Project window); this creates a **prefab**, which is basically a blueprint for Unity to make copies of your item when the game is playing. This will be your item in the AB. Depending on how your FBX was structured, you may make the topmost Game Object into the prefab, or you may use one of its children. Whichever Game Object you drag becomes the root of the prefab; I will refer to it as your item's root.

So select your item's root in the Hierarchy window, and in the Inspector window add Component; select the right MB for your item type (KK Modding Tools has all these MBs, otherwise you'd have to edit the AB with SB3U to copy the MB from a different item like the old days). On the MB change the size of the RendNormal array to however many Mesh Renderers the item has, and then drag those Game Objects from the Hierarchy window into the slots. Fill in any other MB settings. If your item needs other MBs like DynamicBone, add them here and fill them out also. There's a cog wheel in the upper right of the Component for copying and pasting from other items, such as copying DynamicBone settings from the Accessory Hair Example, but you will need to fill in the Root slot with the proper bone from your item.

Your item should look decently close to how it will look in-game because of the preview shaders (except clothes because the shader depends on the game to generate MainTex), but you can also see it with the Day/Sunset/Night Scene Filters. Select the camera in the Hierarchy window, and in the Amplify Color Effect Component (in the Inspector window) you will see a slot for a Lut Texture. You can drag one in from the Project window in Assets/Common/LUT. The filter will probably be applied in both the Game window and the Scene window.

So hopefully the item looks about how you want it, otherwise go back and change things. You generally want the Transform of the prefab's root to be "identity" (no change), so you should check to see if positions and rotations are all 0, and scales are all 1. Your item may work anyways so you should still finish building the mod, but if it has problems with position/rotation/scale there's a good chance it's this. Sometimes you can fix it in Unity, sometimes you must go back to your modeling software and fix it before export. Rotation problems are common because in Blender Z axis is up, but in Unity Y axis is up. But anyways, make the prefab (as described earlier).

Now that you have the prefab (in the Project window), delete the whole thing from the Hierarchy window. Assign the prefab to a new AB for this mod. The material and textures on it will automatically be put in the AB, but if you're doing clothes you must manually assign the MainTex and ColorMask. Assign anything else your mod needs to an AB.

Do the whole list file thing, referencing your prefab. For Studio Items, list files are a bit different and must be in a folder under List/Studio; see the Studio Item Example. Then you're good to build ABs and build the zipmod.

If your item has problems that you think you can fix without redoing the whole process, drag the prefab back into the Hierarchy window (this creates an “instance” of the prefab). Edit it, then hit the Apply button in the upper right corner of the Inspector window (which saves changes to the prefab). Delete the prefab instance from the Hierarchy window, and rebuild ABs and the zipmod.