Modding Hellish Quart Documentation

How mods work in Hellish Quart:

To make mods for Hellish Quart you will need to export your mod files as Asset Bundles using Unity engine. Hellish Quart will recognize and use them in the game. This is very easy and doesn't require coding.

Currently mods are being spawned in Hellish Quart as custom character skins. Character selection screen will detect if there are mod files for a chosen character, and you will be able to choose the mods as character skin customization.

Along with the character skins, or even instead of them, you can include anything you like in the mod – static environment objects, particles, PlayMaker FSMs, music files, sounds, separate animated characters (for example chickens walking next to characters), etc.

1.

How to prepare

1. To prepare a mod for Hellish Quart, please download and install Unity engine on your PC. The Unity 2019.4.0f1 LTS is recommended, other versions might not be compatible with the game. You can download it from https://unity.com/download.

2. Create a new Project, choose the 3D rendering pipeline (not URP and not HDRP).

3. Install the Asset Bundle Browser package. You can import it through Package Manager inside Unity, or download it from GitHub.

4. Download the Hellish Quart Modding Kit from here and import it to your Project.

You are now set up to create mods!

2.

Currently supported mods

Character Skins:

Your most typical mod would be a character skin – a character model that will replace the game's original character model. For example, it could be a samurai instead of Isabella. To do that, you will have to have a samurai 3d model that is skinned to the Hellish Quart Skeleton (provided in the Hellish Quart Modding Kit).

If you don't want to skin the samurai model to the Hellish Quart Skeleton (usually it is done in a 3d editing program like Blender or Maya), and your samurai is already skinned to some other skeleton, you can also just parent each samurai bone to the corresponding bone of the Hellish Quart Skeleton (for example: pelvis to Hips, l_thigh to LeftUpLeg etc.).

If you name this mod file "Skin", the original character will be hidden and the samurai will render instead.

Character and other Props:

Another typical mod is a character prop, for example a hat model, or a backpack model, that will spawn on the existing original Hellish Quart character.

For example, to make Kalkstein wear a hat, you should parent the hat model (it doesn't have to be skinned, but can be if you want) to the Head bone of the Hellish Quart Skeleton (provided in the Hellish Quart Modding Kit).

If you name this mod file "Prop" the original character will still render, but the hat will be added to it.

Sword models:

You can also replace the weapon model that your character wields. Make sure your custom sword model has a Pivot and rotation that is aligned with a sword template, that you can find in Hellish Quart Modding Kit. You don't have to skin the sword to any bones. Just name it "Sword".

If you name this mod file "Sword", the original character's weapon will be hidden and your custom sword model will render instead.

You can also have 1 Skin and 1 Sword in 1 mod file. Or you can have 1 Prop and 1 Sword in the mod file. Please do not put Skin and Prop in 1 mod file – in that case only Prop will spawn.

So for example, if you put Skin and Sword in your mod file, the character skin will change and the sword too.

What can you add to the mods?

- GameObjects, for example Cubes, Spheres, Empty GameObjects, Planes, Quads etc.
- 3D models, for example hats, backpacks, gloves, or even entire houses
- **Skinned 3D models**, for example character skins, or even entire separate animated characters, like chickens walking next to the player
- Light Sources, for example point lights or spot lights
- AudioSources and audio clips
- Particle Emitters
- **PlayMaker** FSMs (Hellish Quart supports Playmaker v1.9.5)
- **Cinemachine** Cameras that change the game's view (enable them by your own attached script after spawning)
- UI elements

Now, you will create two AssetBundles in the Asset Bundle Browser and drop your Skin or Prop or/and Sword files to one of them, and your Skin_pic file to the other.

3.

Naming convention

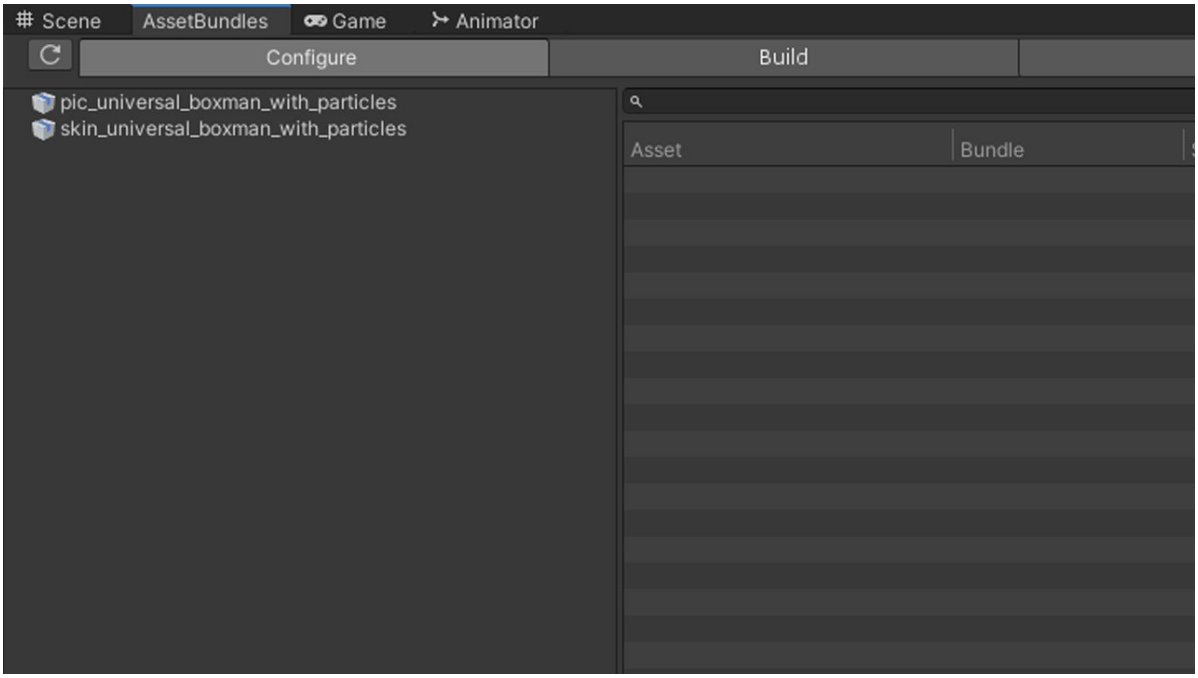
How to name the final Asset Bundle files when you export them with the Asset Bundle Browser:

For universal skins (that will be detected and fit all characters) the Asset Bundle files must be named:

pic_universal_[modname]
skin_universal_[modname]

Example (you need both files for the mod to work):

pic_universal_awesomecharacterskin
skin_universal_awesomecharacterskin



For skins for a specific character (that will be detcted only that character) the Asset Bundle files must be named:

pic_[charactername]_[modname]
skin_[charactername]_[modname]

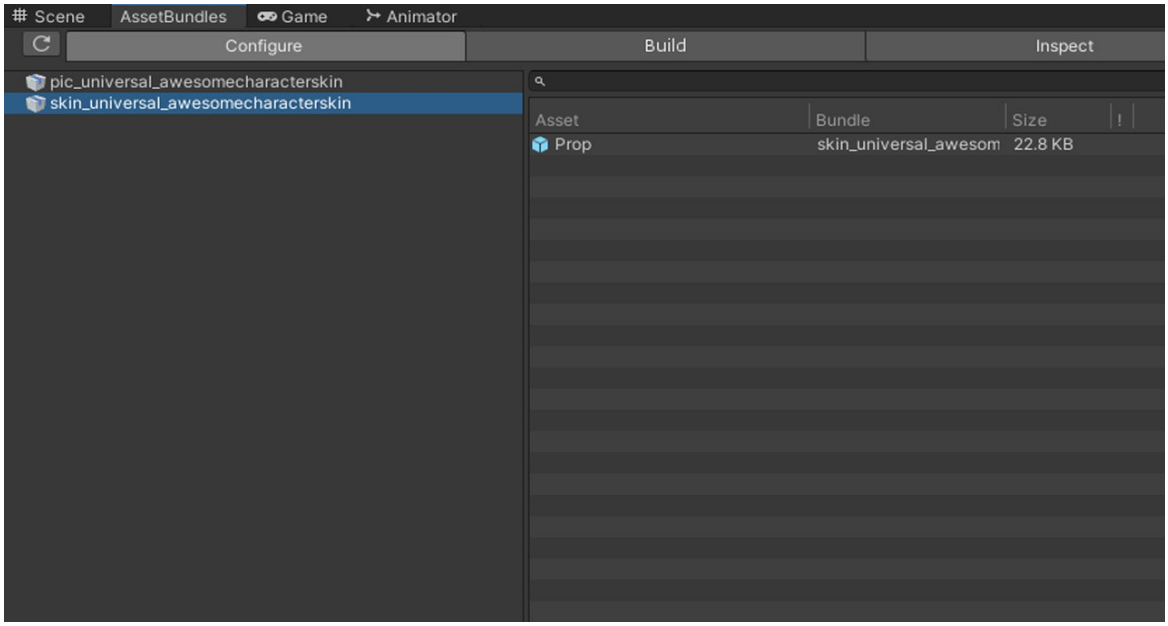
Example (you need both files for the mod to work):

pic_gedeon_awesomecharacterskin
skin_gedeon_awesomecharacterskin

How to name the Prefab or GameObjects that go inside the Skin Asset Bundle:

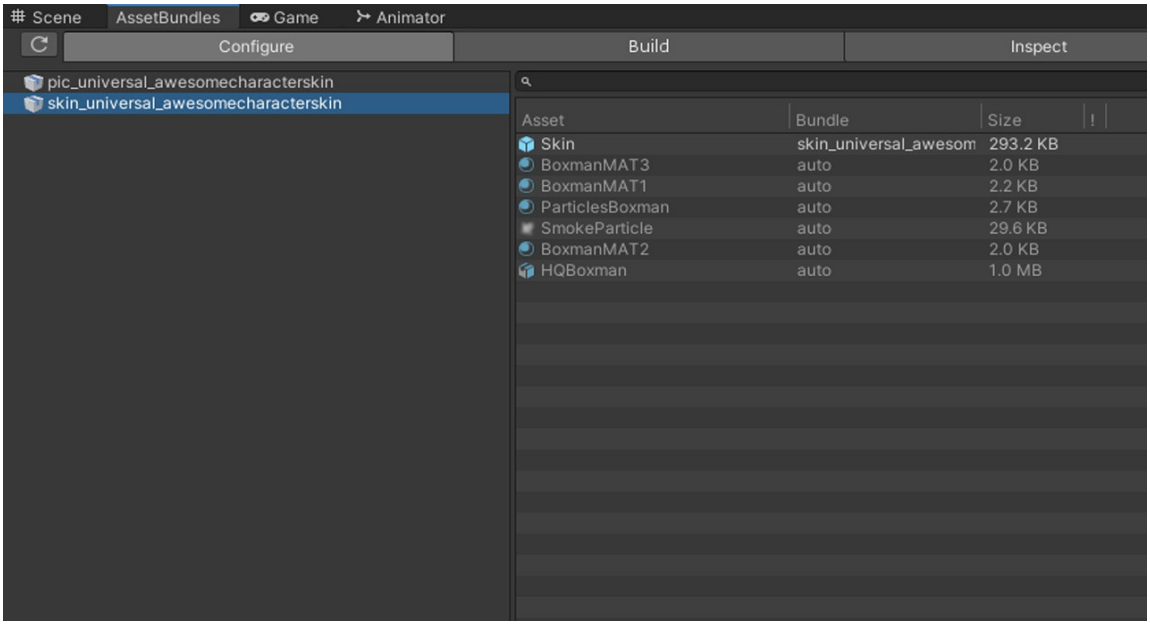
The GameObject or Prefab inside the *skin_universal_[modname]* or *skin_[charactername]_[modname]* AssetBundle must be named (choose only one!):

Prop – if you name your skin model "Prop" then the original character skin will not be replaced, and this file will be added on top of original model.



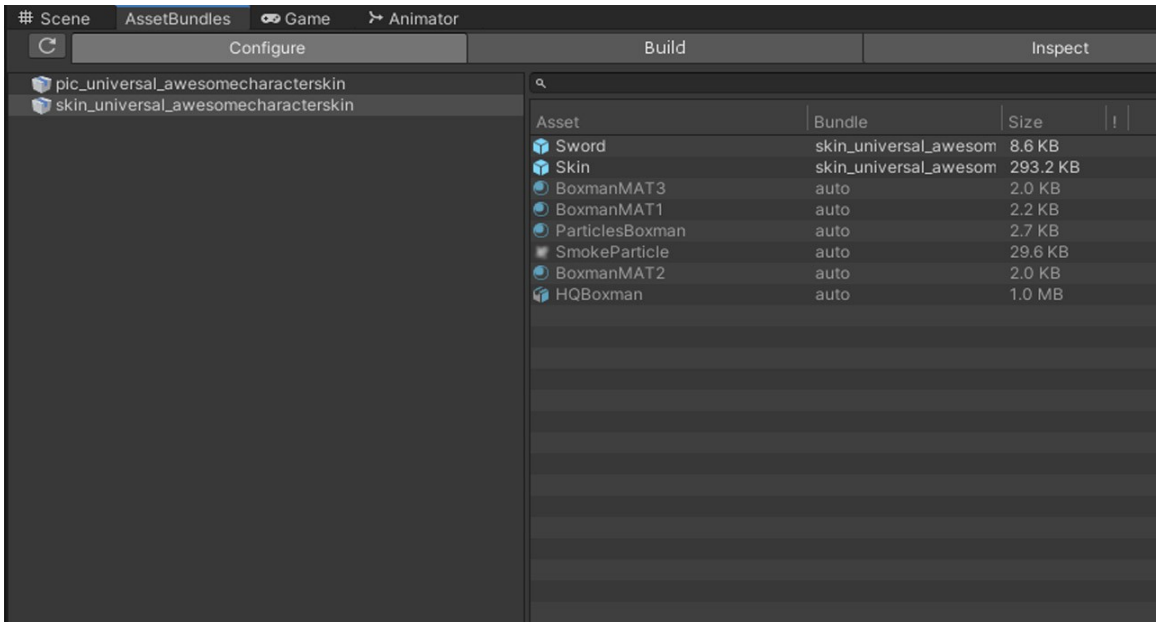
Important: Note that only the GameObject is named Prop, but the AssetBundle file is still named skin_universal_awesomecharacterskin and NOT prop_universal_awesomecharacterskin.

Skin – If you name your skin model "Skin", the the original character mesh will be hidden, and only your skin will be visible.

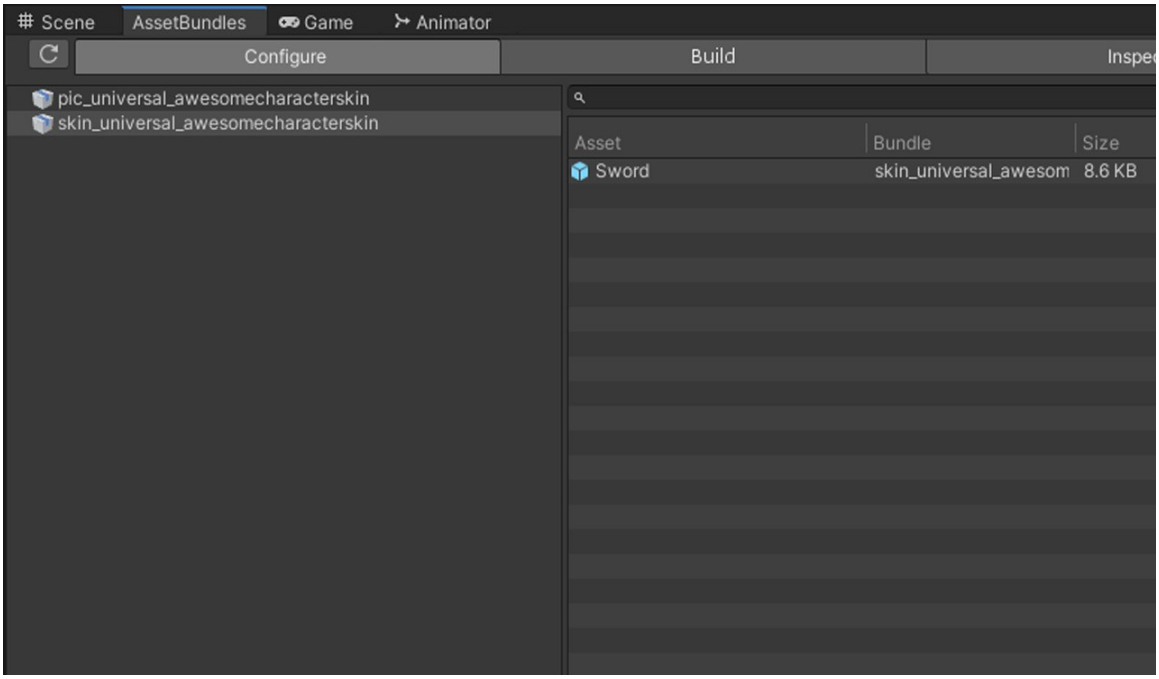


Additionally, to choosing one of the above GameObjects or Prefabs, you also can add another GameObject or Prefab for the sword skin (if you want). It should be named:

> **Sword** – if this GameObject is in the skin AssetBundle file, the original sword model will be hidded and this model will render instead.



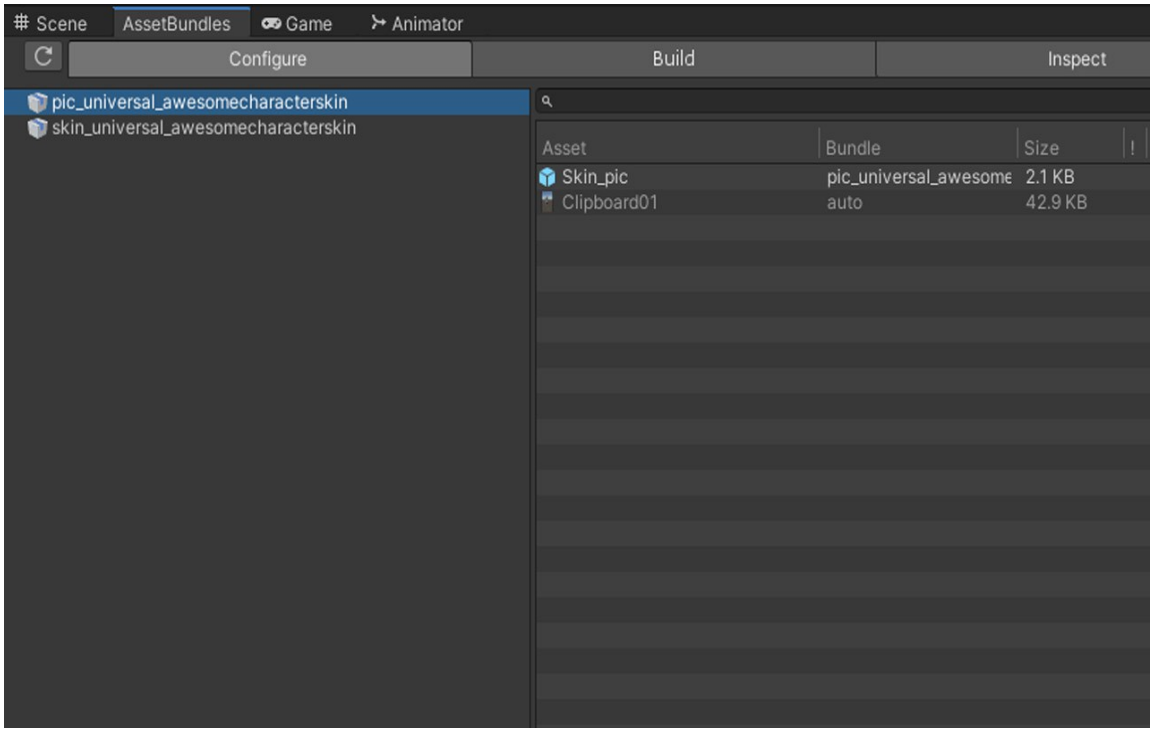
If you wish, you can also not include Prop or Skin in the AssetBundle, and only include Sword. This way only the sword will be modded. But remember to always export a Skin_pic bundle for that sword!



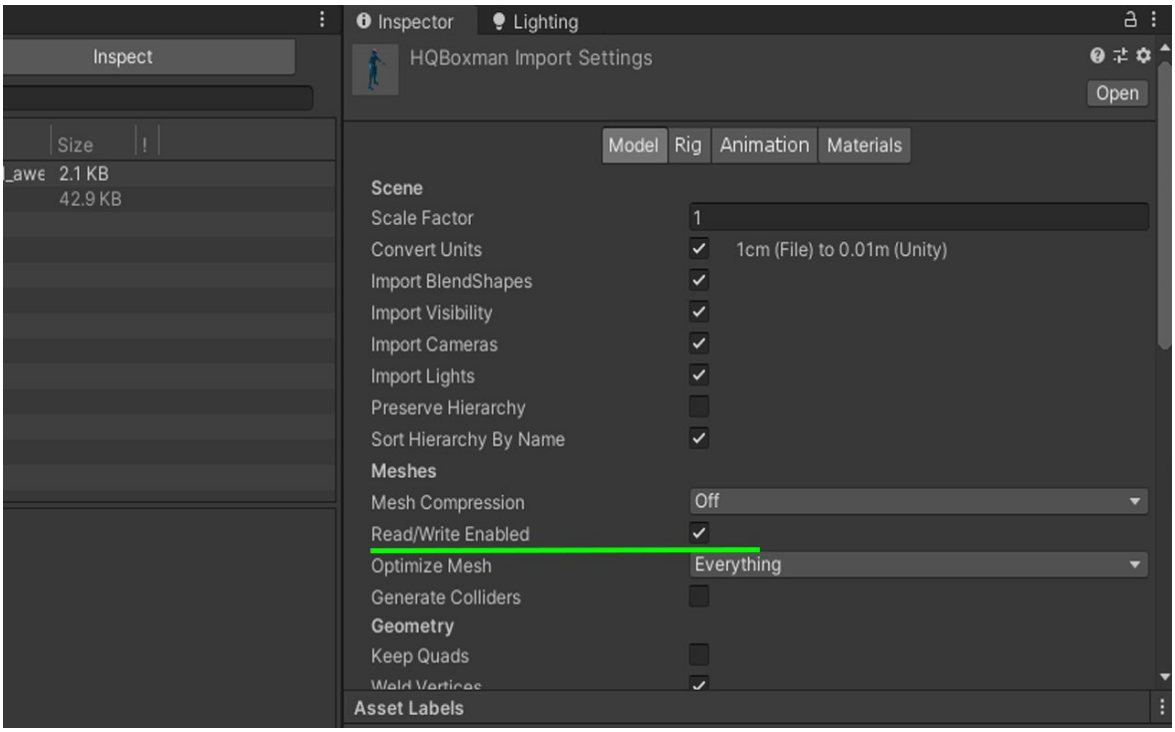
I advise making sword mods only for specific characters and not universal, because every character has a different length of the weapon, or even completly different mechanics (Yendrek).

The Prefab inside the *pic_universal_[modname]* or *pic_[charactername]_[modname]* mod must be named:

Skin_pic – don't change this GameObject's name (it should be named Skin_pic even if the mod GameObject is named Prop or Sword). This GameObject is only for providing a picture of your mod in the Character Select Menu. This object is supposed to be exported as a separate AssetBundle file. **This Object is mandatory for every mod - Skin, Prop or Sword!**



IMPORTANT:
Skin models and meshes source files MUST be marked as *Read/Write Enabled* (Checkbox In the Inspector, in the Model tab – the default is off, so be sure to turn it on manually). Otherwise the game Might break/crash or go into infinite loop.



IMPORTANT:
GameObject names should not have special signs in them and no spaces. For example, when you create a particle, Unity might name it "Particle System". Rename it to "ParticleSystem", wihout the space, or this object will not spawn in the game.

IMPORTANT:
Sword models should not have any Colliders on them, if you want the sword collisions to work as intended.

Once you exported your mod, it should consist of 4 files: the "skin_" and the "pic_" files, and another 2 files named the same with the .manifest extension . So **one mod always has 4 files**.

For example:
pic_universal_awesomecharacterskin
pic_universal_awesomecharacterskin.manifest
skin_universal_awesomecharacterskin
skin_universal_awesomecharacterskin.manifest

So to further explain:

pic_universal_awesomecharacterskin – this file should contain only the Skin_pic GameObject, that has the picture of the mod that will be displayed in the Character Selection Screen.

pic_universal_awesomecharacterskin.manifest – this is just an automatically generated manifest for the file above. Include it in the mod.

skin_universal_awesomecharacterskin – this file should contain your Skin or Prop GameObject. It can also contain Sword GameObject, on top, or even instead Skin or Prop.
skin_universal_awesomecharacterskin.manifest – this is just an automatically generated manifest for the file above. Include it in the mod.

4.

Using the mods in the game

Copy the mod files to:

...[Hellish Quart Install Folder]\Hellish Quart_Data\StreamingAssets

For example:

C:\SteamLibrary\steamapps\common\Hellish Quart\Hellish Quart_Data\StreamingAssets (this is just an example, you may have installed your game copy on some other drive or folder).

If everything was done right, you will see your mod in the Character Selection Menu in the game, when you select a character with the Custom Skin button (X on the Xbox Game Pad).

Tip: Hellish Quart is compatible with PlayMaker, you can attach Playmaker FSMs to your mods as well as scripts.

Tip: Name your mod files using long, original names, to avoid duplicating other mods made by other people. For example, don't name it *skin_isabella_mymod*. Make a unique name instead, for example, *skin_isabella_coolmodders_purplecottonshirt_01*. There is a very low chance that someone else will name their mod like you did. But, if there is a situation that the mod file names indeed duplicate, don't worry: just rename them (but keep the naming convention rules).

Check out our Youtube channel for modding tutorials!

TUTORIALS:

Tutorial 1 - Setting up for Mod creation

1. Download Unity Hub from https://unity.com/
2. Install Unity 2019.4.0f1
3. Create a new prject (3D pipeline)
4. In the top menu open File -> Build Settings... and set up Target Platform to Windows and Architecture to x86_64
5. Go to Window -> Package Manager and import AssetBundle Browser
6. Download and import Hellish Quart Modding Kit from https://www.hellishquart.com/modding
7. Open the Asset Bundle Browser Window (Top menu: Window -> Asset Bundle Browser) and select Build Tab. Set the Build Target to Standalone Windows 64 in the dropdown list.

   All set up!

Tutorial 2 - Creating a basic mod

1. Make a new folder for your mod
2. Open the ModdingScene
3. Create a Cube in the scene (Top menu: GameObject -> 3D Object -> Cube) and parent it to left ForeArm of the Skin skeleton, which you can find in the Scene. Notice that the Cube has a Collider which you can remove, but we will keep it, so it acts like a functional shield.
4. Drag and Drop this Skin to your Mod folder in the Project window to create a new Prefab. If this prefab is called Skin, the original character model will be hidden, but if you name it Prop, it will not be hidded. So in this case we will rename the Skin to Prop, because we want the Cube to render on top of the original character model, not instead.
5. Copy the Skin_pic GameObject from any of the ExampleMods folders and move it to your Mod folder.
6. Using any image editing program, make a picture that will represent your mod in the Character Selection Screen. Import it to Unity and set it as Sprite.
7. Click on the Skin_pic and drag and drop your picture to the image filed in the Inspector.

   Ok, you should now have 3 objects in your folder: Prop, Skin_pic and the picture file. Now let's export the mod.

8. Open the AssetBundle Browser Window (Top menu: Window -> Asset Bundle Browser) and select Configure Tab.
9. Select Prop and Skin_pic (hold Ctrl and click Prop and then Skin_pic) and drag and drop them to Configure Tab window. Choose to create 2 Bundles.
10. Rename the Prop bundle to: skin_universal_[your name of the mod], for example skin_universal_boxpretendingtobeashield
11. Rename the Skin_pic bundle to pic_universal_[your name of the mod], for example pic_universal_boxpretendingtobeashield
12. Go to Build Tab and click Build.

Tutorial 3 - Creating a character skin

1. You will need a new model that will replace the original character model. The easiest way to obtain those is from Asset Store, Turbosquid, cgtrader and similar www sites.

   If you can do it, it is best to skin your new model to the Skin.fbx skeleton file in Blender or Maya and import it back to Unity. Then you proceed like in Tutorial 2, but you name your Prefab Skin, not Prop, so the the original character model will be hidden in the game and only your new skin renders.

2. Easier, but more "hacky" way is to get a character model that is already skinned to some other skeleton. It should have normal human proportions, similar to Hellish Quart's characters.

   What you want to do now, is to drop this character to Scene, and pose it in a way, that it's bones aligns to the bones of Hellish Quart skeleton. Now, using the Hierarchy Tab, parent each of your new character's bones to the corresponding bones of the Hellish Quart Skeleton (example: b_root to Root, pelvis to Hips, l_thigh to LeftUpLeg, l_shin to LeftLeg, l_foot to LeftFoot).

   Make sure the bones of your new character have different names than Hellish Quart skeleton! If they are the same - rename them. Then you proceed like in Tutorial 2, but you name your Prefab Skin, not Prop, so the the original character model will be hidden in the game and only your new skin renders.

VIDEO TUTORIALS on YouTube

Typical problems when creating mods

- I made a character Skin, and it works, but it causes a bug where the fight doesn't end after winning / losing.

  You most likely forgot to mark your skin FBX file as ***Read/Write Enabled*** (Check the box In the Inspector, in the Model tab – the default is off, so be sure to turn it on manually).

- I made a mod but the game doesn't detect it.

  Possible causes and fixes:

  - A typo in the mod files names
  - Forgetting to add a *character name* or *universal* in the mod files naming (read the section 3 of the documentation above)
  - Lack of Skin_pic mod file (it is needed, read documentation)

- Some objects I added to the Skin skeleton are not importing at all

  A possible cause is special characters or spaces in the GameObject naming. When you create a new GameObject, Unity often names it with spaces in the name (for example *Particle System*, or *Box (1)*). Remove the spaces from naming (*ParticleSystem*, *Box(1)*).

- I have two different mods, but the files happen to have the same name. What do I do?

  Just rename one of the mods, for example add "1" at the end of every file.

- I have a mod that is just for a specific character, but I think it will fit all characters just as well. Can I quickly make the mod universal?

  Yes, just replace the character name with universal in the mod file names. For example replace *_gedeon_* with *_universal_*. And vice versa: if you think some mod should not be universal, but just for specific characters, just change the mod files naming.

- I made a mod that has some Objects that use the built-in, default Materials of Unity (for example a Particle system), and they just don't render in the game.

  The built-in materials might not be automatically included in the AssetBundle file (the mod file). Please try just to not use built-in Materials. Just make a new Material, so you can see it as an Asset in the Asset Browser, and use it instead.

- My mod is HUGE, it's like 200 Megabytes! How do I make it smaller?

  It's probably the textures format and size. If you use 8k textures for your skin and the files are in PSD, PNG or TGA format, then the mod file will be big.

  Resize the textures in your mod to 1024x1024, or even 512x512, and resave them as .jpg. They should be now under 400kb each, without any visible loss of quality, when you view them on the characters in the game in fight view.

Made by Kubold
Https://www.hellishquart.com