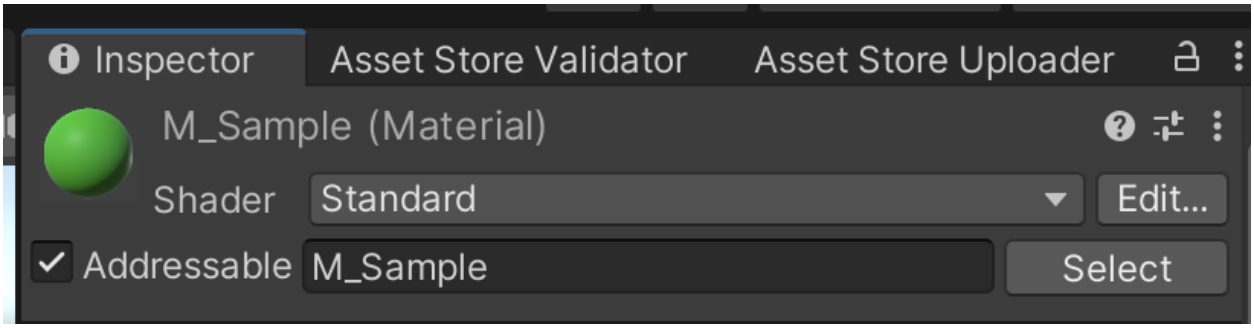
Addresses Manual

Thanks for downloading Addresses!

0️⃣ Pre-requisites

- ⚠️ Your project **MUST** use the Addressables package from Unity
- First, your project must have some Addressable assets configured - you can either drag assets into the window found under `Window > Asset Management > Addressables > Groups`, or you can simply tick the box next to “Addressable” in the Inspector window.

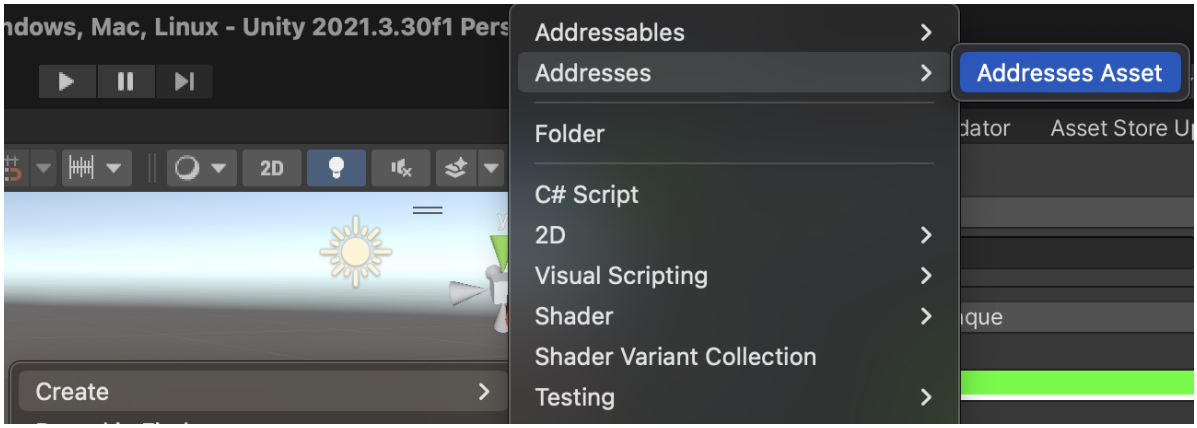


ℹ️ Clicking `Select` here will also open the Addressables Groups window!

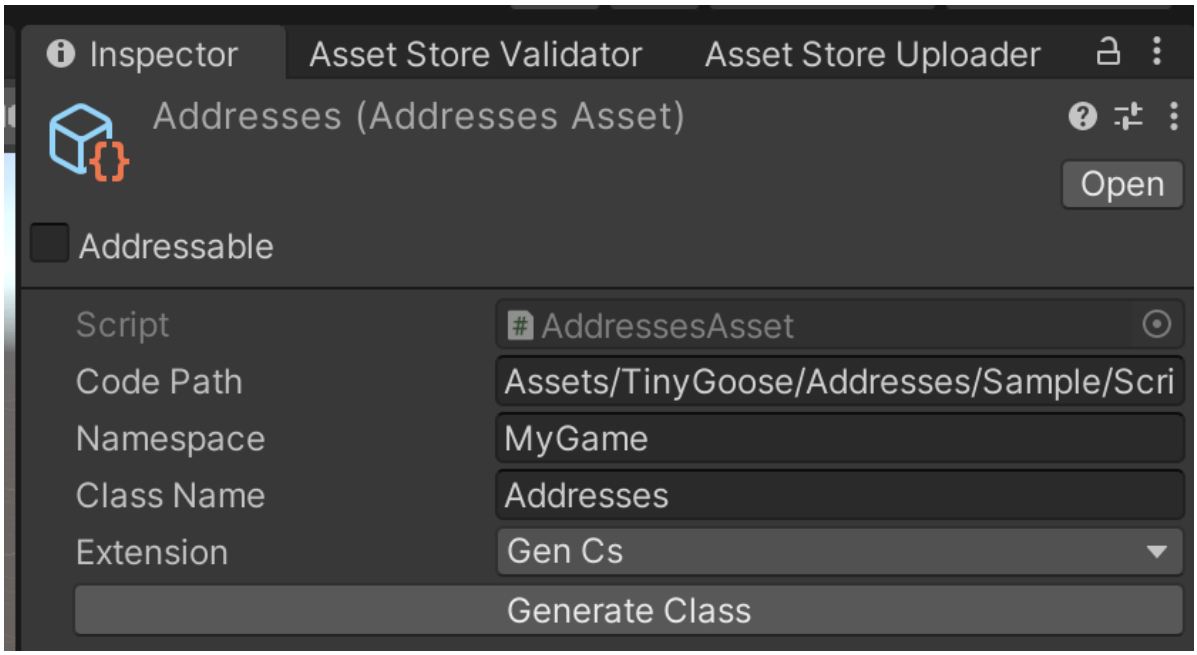
1️⃣ Setting up Addresses

1. You must have installed the package already to be reading this - great job 🎉!

2. First, you need to set up an Addresses asset. This will allow you to configure where the system will place your code-behind file. In the Project window, right-click and choose `Create > Addresses > Addresses Asset`. The name does not matter, and it can be stored wherever you like!



3. In the Inspector, you must now set up where to write your code-behind file. This is for the generated class which will contain the Addresses for all your Addressables.



- **Code Path** - This is where the code file is stored. It should start with `Assets/`. For example, to store a file called `Addresses.gen.cs` at the root of your project, type `Assets/`.

- **Namespace** - If your game uses a namespace, or you want your Addresses class to be in another namespace, fill this in as appropriate. Leave it blank to not use a namespace at all.
- **Class Name** - What to call your code behind class. I recommend leaving this as `Addresses`, but you can use whichever name you please.
- **Extension** - You can choose between a `.cs` extension, or a `.gen.cs` extension (to denote that the file is generated). It makes no difference which you choose (although Unity Editor sometimes doesn't show `.gen.cs` files in the Project window - likely a bug)

4. Finally, click "Generate Class"! Your C# code-behind file is generated with a snapshot of the contents of your Addressables, and you can now begin using the system

2️⃣ Usage

Now that you're all set up, it's time to use Addresses! This is the easy part - whenever you would usually use `Addressables.LoadAssetAsync<...>("some address")`, instead use an entry from your Addresses class. For example, `Addressables.LoadAssetAsync<Material>(Addresses.M_MyMaterial)`. That's it!

♻️ Re-generation

Whenever you add or remove Addressables, you should re-run the code generation to ensure it is up-to-date. You can do this by clicking the "Generate Class" button again from your Addresses asset, but for simplicity you can also use the menu under `Tools > Addresses > Re-generate All`.

📣 Feedback

Doesn't work? Confused? Feedback of any kind?

Feel free to email us at `hello@tinygoose.com` for help and support. I'd also love to hear if you use this package in a shipping game (although crediting is NOT necessary)!