Creating Old World Mods

Updated 14 July 2020.

About this doc:

This doc was written by Dale. If you have any questions I am happy to answer them. You can find me either on the official Old World discord in the #modding channel, or in the Old World Facebook group. See you there!

Old World discord: https://discord.gg/vRNtvW

Old World Facebook: https://www.facebook.com/groups/564557251147563

Modding tools:

For basic modding, you only need some form of text editor to read the XML files. You could use dedicated XML software, but you can easily create an entire mod in Notepad if you like.

For asset and UI modding, you will need Unity 2019.4 LTS. You will also need some way to create asset bundles, whether that be by Unity's asset bundling software or by script in the editor.

I highly recommend having a tool that can extract the game's asset bundles. There are many tools that are able to do this, but I use Asset Studio available from here:
https://github.com/Perfare/AssetStudio/releases

Compiler:

For any sort of DLL modding, you require a compiler. Old World is built on Unity 2019.4 LTS. You can use either Mono or Visual Studio for your environment and compiler.

As the game is built on Unity 2019.4 LTS you can use those Unity docs for API reference.

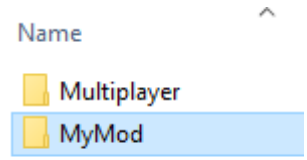
https://docs.unity3d.com/Manual/index.html

The game uses .Net Framework 4 so remember to ensure your environment supports that.

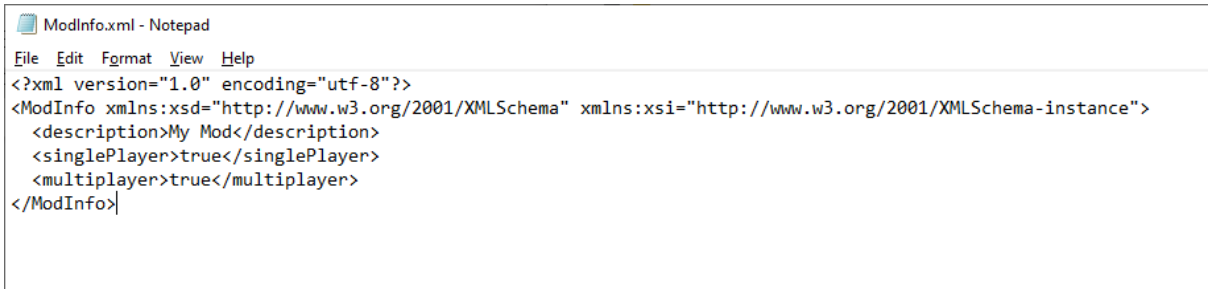
Mod Directory:

Old World supports Mods via mod directories. To create a mod create a folder in the Mods folder such that you have the following path:

<your Windows profile directory>\Documents\My Games\OldWorld\Mods\<Your Mod Name>\



Within your mod directory create an xml file named ModInfo.xml and place the following content into it.



Change the <description> tag to a description of your mod. The mod folder name is what displays in the mod selection menu as the “name” and the description shows when the mod is selected. Save the file.

In game, go to the Mods menu and turn your mod on. Click Save to confirm your mod selections and then start a new game playing the loaded mods.

XML files:

The XML files contain all of the game's data. A mod is able to create new content, change new content, append to existing content and remove content. All XML files go into a folder named Infos within your mod folder. Mods\<Your Mod Name>\Infos\

To add new content the file name must include -add in the name. For instance, a new nation would be added through the file nation-add.xml.

To change content the file name must include -change in the name. For instance, to make changes to Assyria the file is nation-change.xml.

To append an entry to existing content the file name must include -append in the name. For instance, to add your crest sprite sheet to the crests sprite group the file is spriteGroup-append.xml.

To remove content the file name is remove.xml. To remove items you just need to include the Type, not the entire XML class.

```
<Entry>
  <zType>SUCCESSIONORDER_LAWCLASS</zType>
</Entry>
```

Note: The difference between changing or appending an xml file can be summarised as:

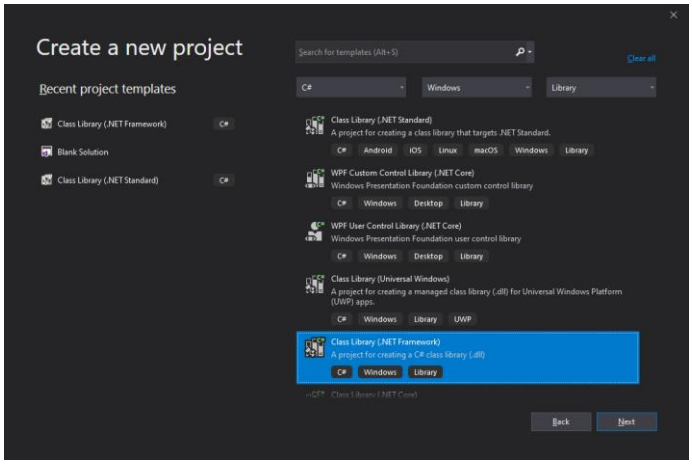
- Changing an xml element allows you to modify existing content and fields.
- Appending allows you to add new lines to an existing xml element.

DLL Modding:

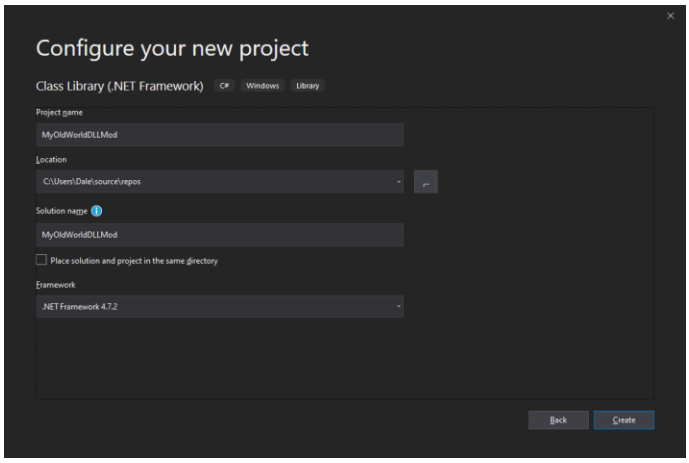
I am going to show you the process to create a DLL mod using my preferred environment, Visual Studio 2019.

Open Visual Studio 2019.

Select Create New Project and pick Class Library (.Net Framework).



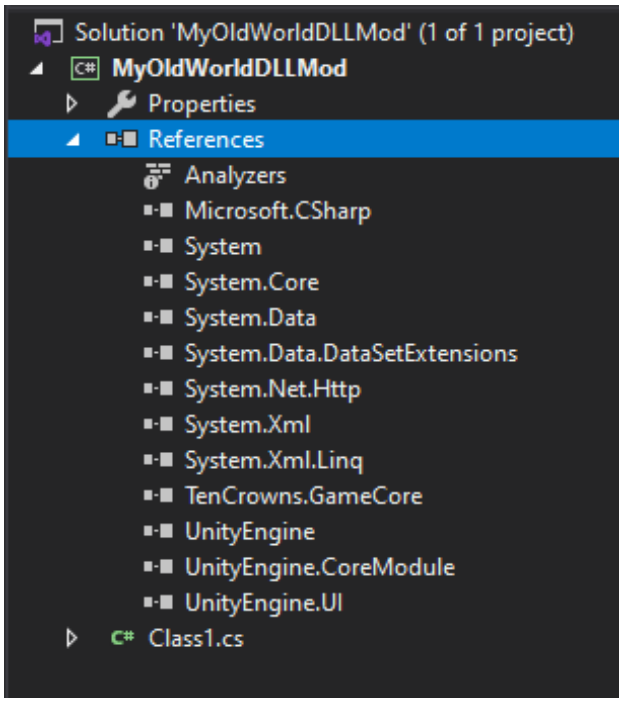
Give the project a name and ensure the Framework option is at least .Net Framework 4.



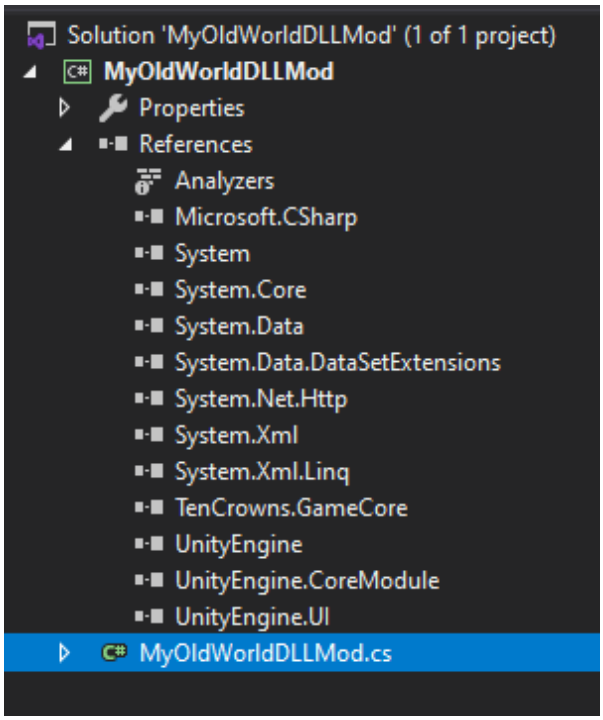
Create the project.

All the reference libraries for Old World are included in the following directory:
\OldWorld_Data\Managed\

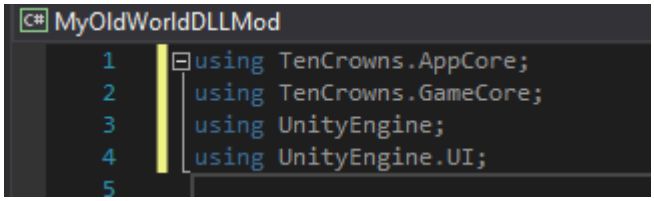
Add UnityEngine.dll, UnityEngine.CoreModule, UnityEngine.UI.dll and TenCrowns.GameCore.dll to the project references.



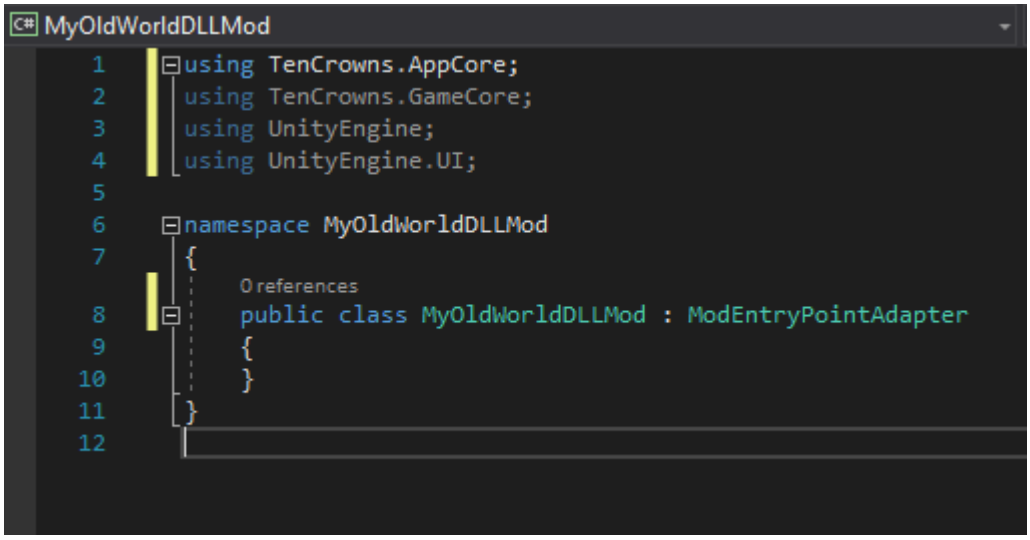
Rename your cs file.



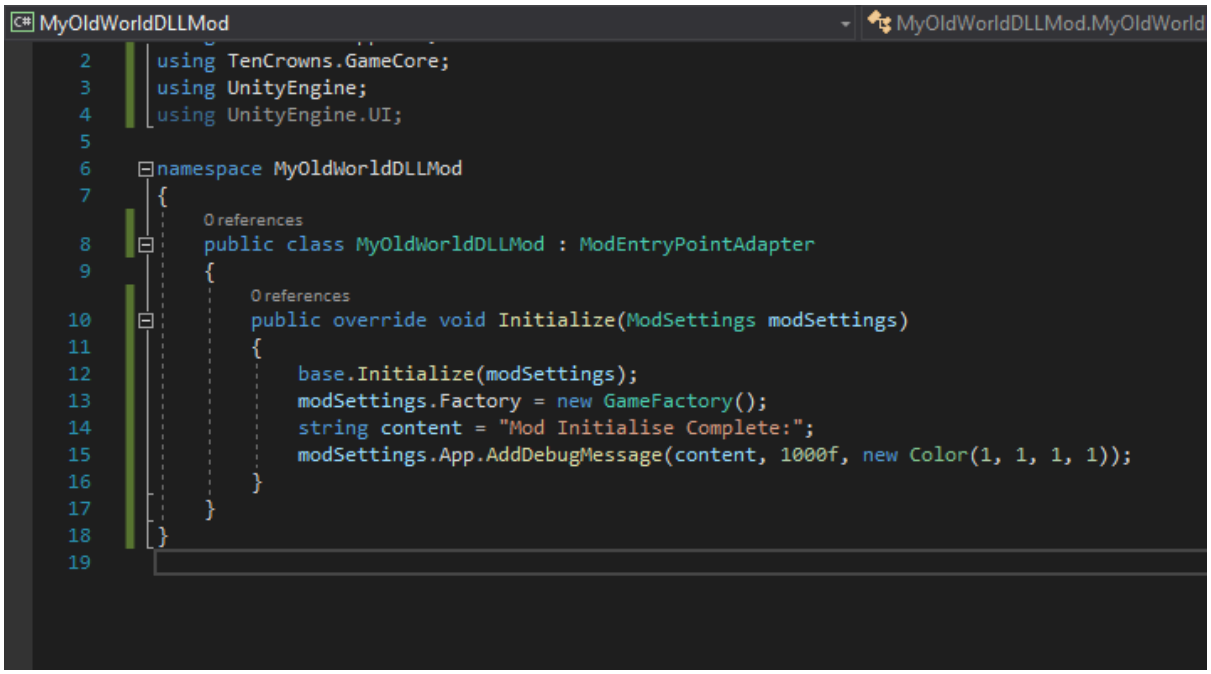
Use the following “using” statements.



Derive your class from the Old World class ModEntryPointAdaptor.

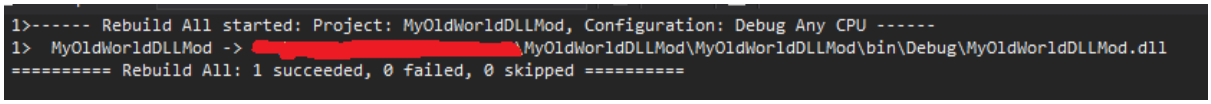


Add your override Initialize() function.

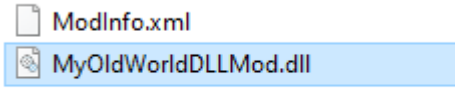


This function contains the basics that all mods require. After referring the base, this function will create a new GameFactory. ModSettings exposes a lot of the game systems and data to your mod. Once the GameFactory is created the message “Mod Initialise Complete” is displayed on screen. This shows your mod has loaded.

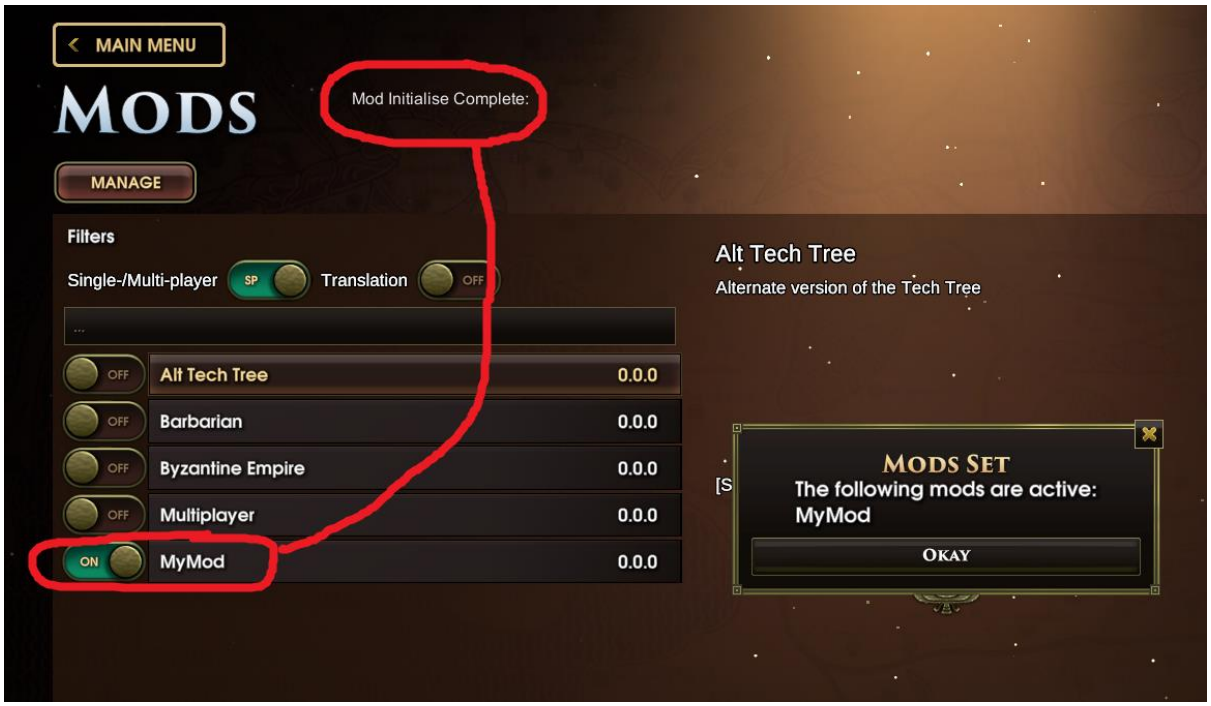
Compile your code.



Copy your DLL file to your mod folder (or use post-build events in Visual Studio).



Now the fun part. Run the game, turn your mod on, and watch for the indicator text to show it is loaded.



And that's it!

Enjoy modding. 😊