

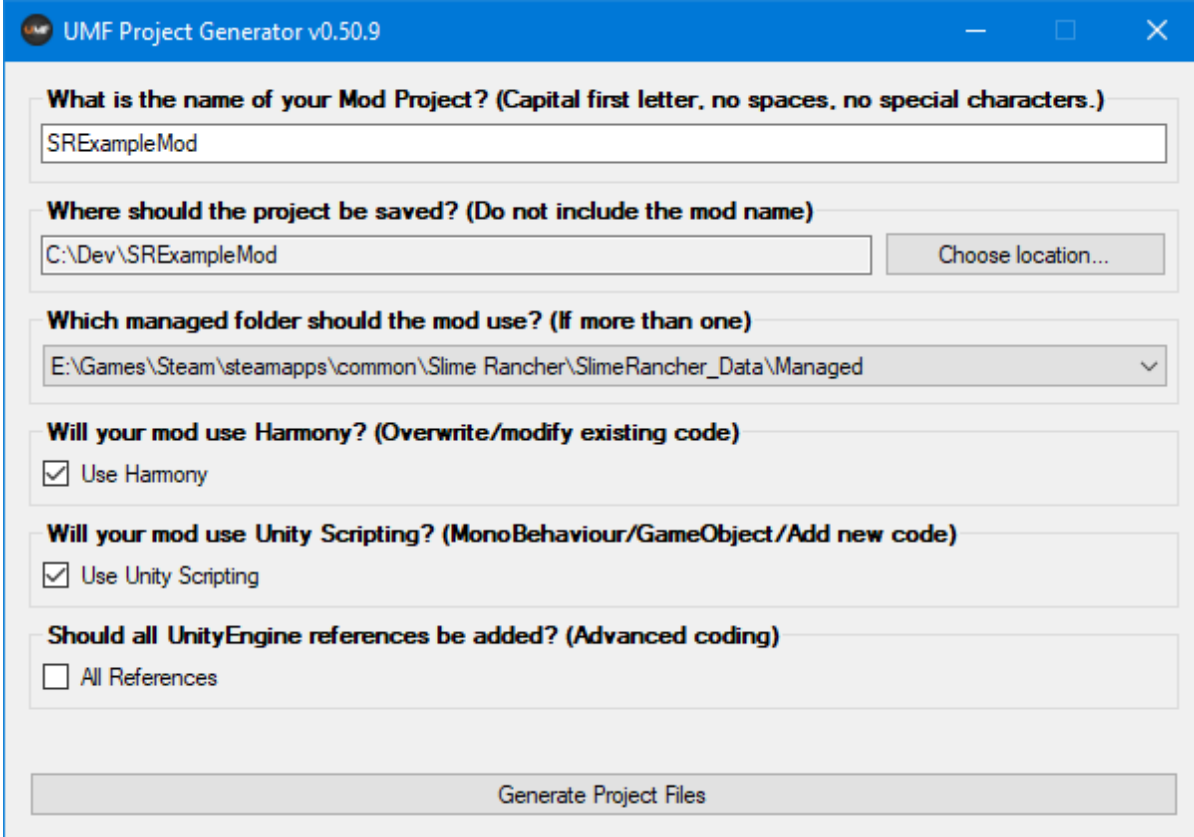
## Modding

# Requirements

- C# Compiler (Recommended: [Visual Studio Community 2017 or Newer](#))
- .NET Decompiler (Recommended: [dnSpy](#))
- **uMod Framework**
- [UMF API Documentation](#)
- [Unity Script Reference](#)
- [Harmony Documentation](#)
- A game with UMF installed into.
- Some experience and/or patience and willingness to learn.

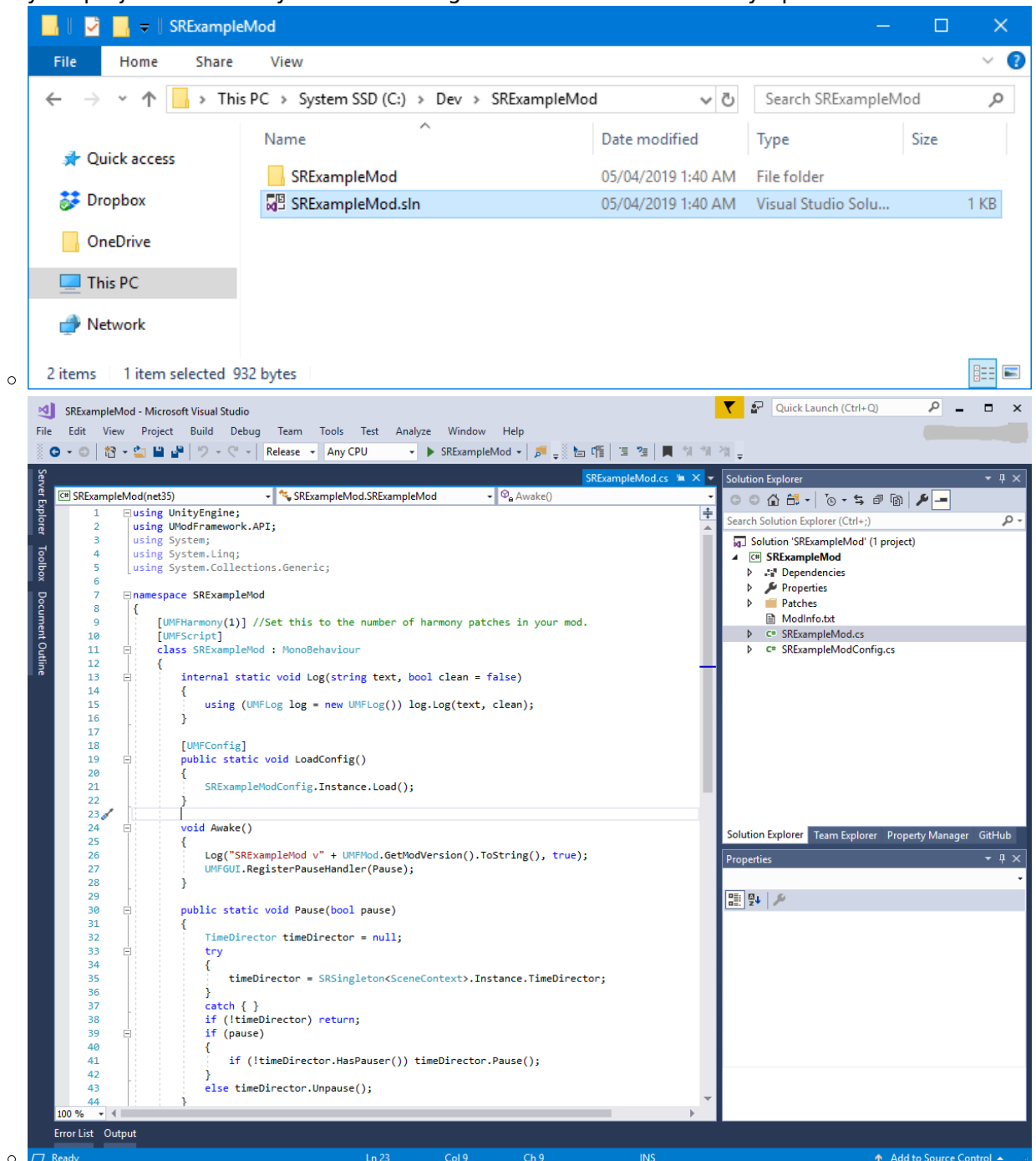
## Mod Project Files

1. Run `UMF.ProjectGenerator.exe` in `\uModFramework\Tools\` and fill in the data, then click Generate Project Files.

The image shows a Windows-style dialog box titled "UMF Project Generator v0.50.9". It contains several input fields and checkboxes for configuring a mod project. The fields are: "What is the name of your Mod Project? (Capital first letter, no spaces, no special characters.)" with the text "SRExampleMod"; "Where should the project be saved? (Do not include the mod name)" with the text "C:\Dev\SRExampleMod" and a "Choose location..." button; "Which managed folder should the mod use? (If more than one)" with a dropdown menu showing "E:\Games\Steam\steamapps\common\Slime Rancher\SlimeRancher\_Data\Managed"; "Will your mod use Harmony? (Overwrite/modify existing code)" with a checked checkbox "Use Harmony"; "Will your mod use Unity Scripting? (MonoBehaviour/GameObject/Add new code)" with a checked checkbox "Use Unity Scripting"; and "Should all UnityEngine references be added? (Advanced coding)" with an unchecked checkbox "All References". At the bottom is a large "Generate Project Files" button.

1. Mod name with no spaces or special characters. This is used for the project files, class and assembly name.
2. Where to save the source code of the mod.

3. If a game has more than one data/managed folder choose the appropriate one. (32bit/64bit/etc)
  4. Harmony is used to overwrite existing code or inject new code into existing functions.
  5. Unity Scripting is used to add new code to the game, the same as the game already uses. See the Unity Script Reference to learn about how MonoBehaviour GameObjects work.
  6. All References adds all dll files in the Managed folder as References, giving you access to all Unity classes as well as any separate classes the game provides. Only use if you know what you are doing.
2. Open your project solution by double clicking ModName.sln in the newly opened folder.



3. Add your code and click Build > Rebuild. (Rebuild will clean your project before each build.)
- See [Decompiling Game Code](#) to learn about finding things to mod in the game code.
  - See [Important Links and Info](#) for more useful info.

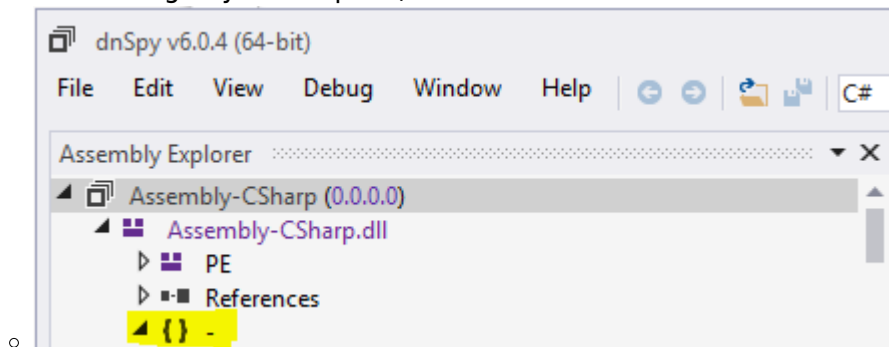
# Decompiling Game Code

In order to modify existing game code or figure out which game code you want to interface with you must decompile the game code using any decompiling program.

Due to the vast superiority of [dnSpy](#) we highly recommend you use it to browse the game code.

Once you have installed dnSpy follow these steps:

1. Run dnSpy once so that it integrates with windows explorer's right click context menu.
2. Navigate to ...\- 3. Right click Assembly-CSharp.dll and choose **Open with dnSpy**.
  - Alternatively you can click **File > Open** inside the dnSpy and open this file.
- 4. Navigate through the **Assembly Explorer** on the left side until you reach the - namespace (Code not having any namespace).



- This is where you will find the vast majority of the game code for most games.
  - Some games may use other namespaces for some of it's code.
5. Use the **Assembly Explorer**, **Search** and **Analyze** tools of dnSpy to discover code relevant to what you want to mod and learn where it is tied to.
    - See [My First Slime Rancher Mod \(Modding Guide\)](#) to learn various ways to create a basic mod.
    - See [Important Links and Info](#) for more useful info.

## Important Links and Info

- **UMF API** - The UMF API gives you access to various functions that makes modding easier, and is also required in order for the mod to be started.
- **Modding Guides** - A list of guides to assist you in creating mods for UMF.
- **Harmony** - Harmony is a useful library provided with UMF that lets you overwrite and inject code in MEMORY into existing classes and functions.
- **Example Mods** - A list of open source UMF mods you can use as examples to better learn

coding with UMF.

- **Source Mods** - UMF can compile and run mods at runtime directly from .cs files.
- **UMF Patch** - An IL-based UMF scripting patch system used as a last resort to let you automatically overwrite code in dll files on disk when Harmony can't access it in any way.
- **Mod Installer** - The UMF Mod Installer (URI Handler) lets you create one click Install links for your mods.
- **UMF Hosting** - This will show you the steps to having your mod added to mod list on this website.

UMF starts and access mods through **Attributes**.

Mods can currently be provided as .cs files, .dll files, .zip files(containing .dll files), and .umfmod files(packed and encrypted) by the **Mod Packer**.

UMF and Mods made with UMF does not re-distribute any code or game files from games or the Unity Engine. Making them both morally and legally safe.

Mods made with UMF can also be freely sold/marketed by the Mod Creator. (Provided they don't violate the UMF License)

From:

<https://umodframework.com/wiki/> - **UMF Wiki**

Permanent link:

<https://umodframework.com/wiki/modcreation>

Last update: **2024/04/24 22:22**

