Modding

My First Slime Rancher Mod (Modding Guide)

This guide will help you create a simple Slime Rancher Mod for UMF that lets you modify your max health.

It will guide you through all the different methods you could use to modify the players max health, along with the method best suited for this particular task.

It is up to you to determine which method is better suited for anything else you might attempt to mod.

Table of Contents

- Preliminary Setup
- Method 1 (Unity Scripting)
- Method 2 (Harmony Postfix)
- Method 3 (Harmony Traverse)
- Method 4 (Harmony Transpiler)
- Method 5 (UMF Patch)
- Method 6 (Optimal)
- Building
- Finalizing

Preliminary Setup

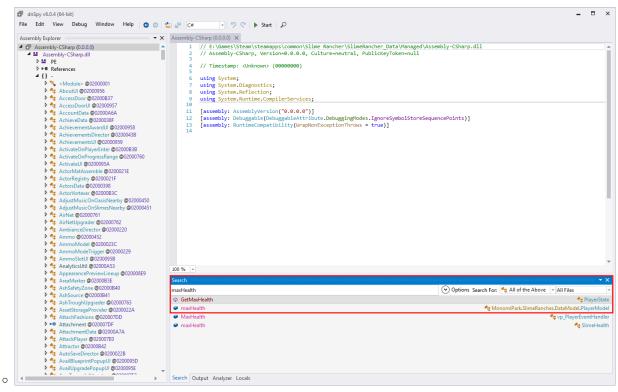
- 1. Start by creating the project files as seen in Mod Creation.
 - $\circ\,$ Use the parameters as seen in the image below.

Where should the project be sav	ed? (Do not include the ma	d name)	
:\Dev\UMF\Mods\MyFirstSRMod			Choose location
Which managed folder should th	e mod use? (If more than o	ne)	
E:\Games\Steam\steamapps\commor	\Slime Rancher\SlimeRancher_	Data\Managed	
Use Harmony Will your mod use Unity Scripting	? (MonoBehaviour/GameO	biect/Add new c	ode)
 Use Harmony Will your mod use Unity Scripting Use Unity Scripting 	? (MonoBehaviour/GameO	bject/Add new c	ode)
Nill your mod use Unity Scripting		-	ode)

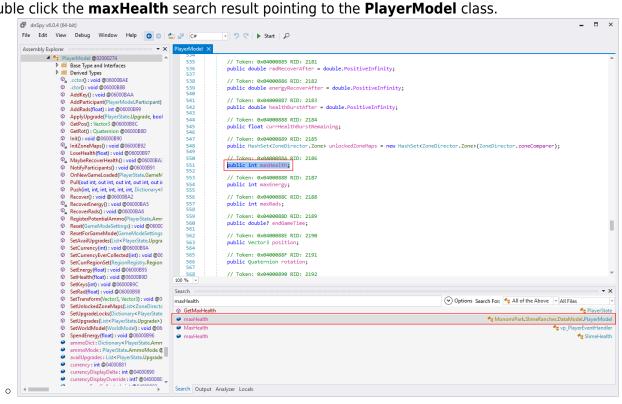
- 2. Open the newly generated solution in Visual Studio.
- 3. Open and browse the game code as seen in Mod Creation: Decompiling Game Code.

	🖆 🔐 C# 🚽 🤊 C 🕨 Start 🔎	
mbly Explorer	× Assembly-CSharp (0.0.0) ×	
Assembly-CSharp (0.0.0.0)	1 // E:\Games\Steam\steamapps\common\Slime Rancher\SlimeRancher Data\Managed\Assembly-CSharp.dll	
Assembly-CSharp.dll	2 // Assembly-CSharp, Version=0.0.0,0, Culture-neutral, Public/EyToken=null	
Assembly-Conarp.dll E		
	4 // Timestamp: <unknown> (0000000)</unknown>	
The second	5 5	
4 {} -	6 using System;	
Module> @02000001	7 using System Diagnostics;	
AboutUI @02000956	8 using System.Reflection;	
AccessDoor @02000B37	9 using System.Runtime.CompilerServices;	
🕨 🔩 AccessDoorUI @02000957	10	
🕨 🔩 AccountData @02000A6A	<pre>11 [assembly: AssemblyVersion("0.0.0.0")]</pre>	
🕨 🔩 AchieveData @0200038F	12 [assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]	
🕨 🔩 AchievementAwardUI @02000958	<pre>13 [assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)] 14</pre>	
AchievementsDirector @0200043B	A+	
🕨 🔩 AchievementsUI @02000959		
ActivateOnPlayerEnter @02000B3B		
ActivateOnProgressRange @02000760		
ActivateUI @0200095A		
ActorMatAssemble @0200021E		
ActorRegistry @0200021F		
ActorsData @02000398		
ActorVortexer @02000B3C		
AdjustMusicOnOasisNearby @02000450		
AdjustMusicOnSlimesNearby @02000451		
AirNet @02000761		
AirNetUpgrader @02000762		
AmbianceDirector @02000220		
Ammo @02000452		
AmmoModel @0200023C		
AmmoModeTrigger @02000229		
AmmoSlotUI @0200095B		
AnalyticsUtil @02000958		
Analyticsotil @02000A55 Analyticsotil @02000A55 Analyticsotil @02000A55	100 % =	
AppearancePreviewEnreup @02000829 AreaMarker @02000B3E	Search	
AreaMarker @02000B3E AshSafetyZone @02000B40		
AshSaretyZone @02000840 AshSource @02000841	🕑 Options Search For: 🔩 All of the Above 👻 All Files	
AshSource @02000841 AshTroughUpgrader @02000763		
AssetStorageProvider @0200022A		
AttachFashions @020007DD		
Attachment @020007DF		
AttachmentData @02000A7A		
AttackPlayer @020007E0		
Attractor @02000B42		
AutoSaveDirector @0200022B		
🕨 🔩 AvailBlueprintPopupUI @0200095D		
AvailUpgradePopupUI @0200095E	* I	

4. Use the search bar in dnSpy to search for maxHealth.



- Here we can see that the max health is retrieved by the **PlayerState** through the GetMaxHealth function for reading purposes.
- Double clicking GetMaxHealth will also reveal that it retrieves the real maxHealth value from the **PlayerModel**, just as seen in the second search result.
- Double click the maxHealth search result pointing to the PlayerModel class.



- Here we can see that maxHealth is stored as a 32bit int.
- 6. Right click **maxHealth** and select **Analyze**.

y Explorer 🔹 🔻 🖡	del ×	
🔺 🔩 PlayerModel @02000274 🔷	// Token: 0x04000885 RID: 2181	
Base Type and Interfaces	<pre>public double radRecoverAfter = double.PositiveInfinity;</pre>	
Derived Types	public double ranketoverancer - double.PositiveInfinity;	
♀ .cctor(): void @06000BAE	// Token: 0x04000886 RID: 2182	
O .ctor(): void @06000B8B	public double energyRecoverAfter = double.PositiveInfinity:	
AddKey(): void @06000BAA		
AddParticipant(PlayerModel.Participant)	// Token: 0x04000887 RID: 2183	
AddRads(float) : int @06000B99	<pre>public double healthBurstAfter = double.PositiveInfinity;</pre>	
ApplyUpgrade(PlayerState.Upgrade, bool	// Token: 0x04000888 RID: 2184	
GetPos(): Vector3 @06000B8C	public float currHealthBurstRemaining:	
GetRot(): Quaternion @06000B8D	public float currhealthourstkemaining;	
Init(): void @06000B90	// Token: 0x04000889 RID: 2185	
Q InitZoneMaps() : void @06000B92	<pre>public HashSet<zonedirector.zone> unlockedZoneMaps = new HashSet<zonedirector< pre=""></zonedirector<></zonedirector.zone></pre>	.Zone>(ZoneDirector.zoneComparer);
CoseHealth(float): void @06000B97		
MaybeRecoverHealth(): void @06000BA;	// Token: 0x0400088A RID: 2186	
NotifyParticipants(): void @06000B91	public int maxHealth;	
OnNewGameLoaded(PlayerState.Gamel/		
Pull(out int, out int, out int, out int, out int	// Token: 0x0400088B RID: 2187	
Push(int, int, int, int, int, int, Dictionary<)	<pre>public int maxEnergy;</pre>	
Recover(): void @06000BA2	// Token: 0x0400088C RID: 2188	
Pe RecoverEnergy() : void @06000BA5	public int maxRads;	
P. RecoverRads() : void @06000BA6	public inclusion,	
RegisterPotentialAmmo(PlayerState.Amr	// Token: 0x0400088D RID: 2189	
Reset(GameModeSettings) : void @06000	<pre>public double? endGameTime;</pre>	
ResetForGameMode(GameModeSettings		
SetAvailUpgrades(List <playerstate.upgra< td=""><td>// Token: 0x0400088E RID: 2190</td><td></td></playerstate.upgra<>	// Token: 0x0400088E RID: 2190	
SetCurrency(int) : void @0600089A	public Vector3 position;	
SetCurrencyEverCollected(int) : void @0600059A	// Token: 0x0400088F RID: 2191	
	public Quaternion rotation;	
	public quaternion rotation,	
© SetEnergy(float) : void @06000895	// Token: 0x04000890 RID: 2192	
SetHealth(float): void @0600089D		
SetKeys(int) : void @0600089C SetRad(float) : void @06000898		
	IonomiPark.SlimeRancher.DataModel.PlayerModel.maxHealth : int @0400088A	<u> </u>
SetUnlockedZoneMaps(List <zonedirectc< td=""><td>Assigned By</td><td></td></zonedirectc<>	Assigned By	
SetUpgradeLocks(Dictionary <playerstate< p=""></playerstate<>	MonomiPark.SlimeRancher.DataModel.PlayerModel.ApplyUpgrade(PlayerState.Upgrade, bool): void @06000B	4B
SetUpgrades(List <playerstate.upgrade>)</playerstate.upgrade>	MonomiPark.SlimeRancher.DataModel.PlayerModel.Reset(GameModeSettings) : void @06000BA7	
SetWorldModel(WorldModel) : void @06	Read By	
SpendEnergy(float) : void @06000B96		
ammoDict : Dictionary < PlayerState.Amm		
ammoMode : PlayerState.AmmoMode @		
availUpgrades : List <playerstate.upgrade< p=""></playerstate.upgrade<>		
currency : int @04000881		
currencyDisplayDelta : int @04000890		
currencyDisplayOverride : int? @0400088:		

- Here we see that **maxHealth** is assigned by **PlayerModel.ApplyUpgrade** and **PlayerModel.Reset**.
- Further analyzing of ApplyUpgrade and Reset will teach you that ApplyUpgrade is used by the in-game upgrade console to apply upgrades the player has purchased, including max health upgrades. The Reset function is used when a save is loaded or when a new game is started and also resets the max health.
- If we look at the **Reset** function we see that the default health without any upgrades is 100.
- If we look at the **ApplyUpgrade** function we see that **maxHealth** is 150, 200, 250, and 350 depending on the health upgrade the player has.
- We have now learned that **maxHealth** is of the **int** type and that **ApplyUpgrade** and **Reset** will modify the **maxHealth**.
- 7. Create a MaxHealth config setting in MyFirstSRModConfig.cs with the following code.
 - Add the following line below

//Add your config vars here.

public static int MaxHealth;

 $\circ\,$ Add the following line below

//Add your settings here

```
MaxHealth = cfg.Read("MaxHealth", new UMFConfigInt(999, 1, 9999),
"This is the player's max health.");
```

• Your config class should now look something like this:

```
using System;
using UModFramework.API;
```

```
namespace MyFirstSRMod
{
    public class MyFirstSRModConfig
    {
        private static readonly string configVersion = "1.0";
        //Add your config vars here.
        public static int MaxHealth;
        internal static void Load()
        {
            MyFirstSRMod.Log("Loading settings.");
            try
            {
                using (UMFConfig cfg = new UMFConfig())
                Ł
                    string cfgVer = cfg.Read("ConfigVersion", new
UMFConfigString());
                    if (cfgVer != string.Empty && cfgVer !=
configVersion)
                    {
                        cfg.DeleteConfig(false);
                        MyFirstSRMod.Log("The config file was
outdated and has been deleted. A new config will be generated.");
                    }
                    //cfg.Write("SupportsHotLoading", new
UMFConfigBool(false)); //Uncomment if your mod can't be loaded
once the game has started.
                    cfg.Read("LoadPriority", new
UMFConfigString("Normal"));
                    cfg.Write("MinVersion", new
UMFConfigString("0.52.1"));
                    //cfg.Write("MaxVersion", new
UMFConfigString("0.54.99999.99999")); //Uncomment if you think
your mod may break with the next major UMF release.
                    cfg.Write("UpdateURL", new
UMFConfigString(""));
                    cfg.Write("ConfigVersion", new
UMFConfigString(configVersion));
                    MyFirstSRMod.Log("Finished UMF Settings.");
                    //Add your settings here
                    MaxHealth = cfg.Read("MaxHealth", new
UMFConfigInt(999, 1, 9999), "This is the player's max health.");
                    MyFirstSRMod.Log("Finished loading
settings.");
                }
            }
```

```
catch (Exception e)
{
    MyFirstSRMod.Log("Error loading mod settings: " +
e.Message + "(" + e.InnerException?.Message + ")");
    }
  }
}
```

You have now added a config setting that can be adjusted from the UMF Menu in-game.
 8. Proceed by testing out all the different methods you can mod the max health below.

Method 1 (Unity Scripting)

This method uses only Unity Scripting and the game's own code to modify the max health value. You can find this method by looking through the game code which will take some experience and analyzing.

For this particular scenario this method is not the best to use.

In Slime Rancher the **PlayerModel** class instance can be accessed through the **GameModel** instance class which in turn can be retrieved from the **SceneContext** using the **SRSingleton** class.

We can set this value directly using the following line:

```
SRSingleton<SceneContext>.Instance.GameModel.GetPlayerModel().maxHealth =
MyFirstSRModConfig.MaxHealth;
```

This would set maxHealth to whatever the config value is set to, in the default case 999. However we have to set this in a way that it is done after **ApplyUpgrade** and **Reset** which is why the other methods are normally better for this case. So to ensure the maxHealth is always set after those we can apply it using Unity Scripting's Update function, which require some extra checks to ensure we are in the game.

1. In your MyFirstSRMod.cs add the following at the top:

using MonomiPark.SlimeRancher.DataModel;

2. Comment out the [UMFHarmony(1)] line.

```
//[UMFHarmony(1)]
```

3. Add the following line to the class:

private static PlayerModel playerModel;

4. Add the following to the bottom of the Awake method:

```
MyFirstSRModConfig.Load();
```

5. Add the following code to the **Update** function.

```
if (!Levels.isSpecial() &&
SRSingleton<SceneContext>.Instance?.GameModel != null) //Makes sure we
are in game and that the GameModel exists.
{
          playerModel =
SRSingleton<SceneContext>.Instance.GameModel.GetPlayerModel();
}
if (playerModel != null) //Make sure that the PlayerModel has been
retrieved.
{
          playerModel.maxHealth = MyFirstSRModConfig.MaxHealth; //Set the max
health to our config value.
}
```

6. Your MyFirstSRMod.cs should now look something like this:

```
using UnityEngine;
using UModFramework.API;
using System;
using System.Linq;
using System.Collections.Generic;
using MonomiPark.SlimeRancher.DataModel;
namespace MyFirstSRMod
{
    //[UMFHarmony(1)] //Set this to the number of harmony patches in
your mod.
    [UMFScript]
    class MyFirstSRMod : MonoBehaviour
    {
        private static PlayerModel playerModel;
        internal static void Log(string text, bool clean = false)
        {
            using (UMFLog log = new UMFLog()) log.Log(text, clean);
        }
        [UMFConfig]
        public static void LoadConfig()
        {
            MyFirstSRModConfig.Load();
        }
        void Awake()
        Ł
            Log("MyFirstSRMod v" + UMFMod.GetModVersion().ToString(),
```

```
true);
            UMFGUI.RegisterPauseHandler(Pause);
            MyFirstSRModConfig.Load();
        }
        public static void Pause(bool pause)
        {
            TimeDirector timeDirector = null;
            try
            {
                timeDirector =
SRSingleton<SceneContext>.Instance.TimeDirector;
            }
            catch { }
            if (!timeDirector) return;
            if (pause)
            {
                if (!timeDirector.HasPauser()) timeDirector.Pause();
            }
            else timeDirector.Unpause();
        }
        void Update()
        {
            if (!Levels.isSpecial() &&
SRSingleton<SceneContext>.Instance?.GameModel != null) //Makes sure we
are in game and that the GameModel exists.
            {
                playerModel =
SRSingleton<SceneContext>.Instance.GameModel.GetPlayerModel();
            if (playerModel != null) //Make sure that the PlayerModel
has been retrieved.
            {
                playerModel.maxHealth = MyFirstSRModConfig.MaxHealth;
//Set the max health to our config value.
            }
        }
    }
}
```

- $\circ\,$ This will make sure that your max health is always 999 every single frame regardless of which health upgrade the player has. This is obviously not normally recommended.
- $\circ\,$ This code also ensures the config value takes effect immediately upon applying it in the UMF Menu.
- 7. See Building for the next steps.
 - WARNING: If this is the first method you try, you will also need to comment out Patch_PURPOSEOFPATCH.cs before building.

2025/08/29 09:00

Method 2 (Harmony Postfix)

9/13

This method is the easiest way to do it, but it's still not entirely optimal.

- 1. If you did Method 1 before this method then perform the following steps:
 - 1. Uncomment the UMFHarmony attribute in MyFirstSRMod.cs.

[UMFHarmony(1)]

2. Comment out the extra config load you added to the awake function.

```
//MyFirstSRModConfig.Load();
```

- This is because the UMFHarmony will automatically trigger the UMFConfig attribute before applying the patches in the mod.
- 3. Uncomment everything in Patch_PURPOSEOFPATCH.cs.
- 4. Comment out any code you added with other methods so they do not conflict with what you are currently doing.
- 5. Your MyFirstSRMod.cs should now look something like this:

```
using UnityEngine;
using UModFramework.API;
using System;
using System.Ling;
using System.Collections.Generic;
using MonomiPark.SlimeRancher.DataModel;
namespace MyFirstSRMod
{
    [UMFHarmony(1)] //Set this to the number of harmony patches in
your mod.
    [UMFScript]
    class MyFirstSRMod : MonoBehaviour
    {
        //private static PlayerModel playerModel;
        internal static void Log(string text, bool clean = false)
        {
            using (UMFLog log = new UMFLog()) log.Log(text,
clean);
        }
        [UMFConfig]
        public static void LoadConfig()
        {
            MyFirstSRModConfig.Load();
```

```
}
        void Awake()
        {
            Log("MyFirstSRMod v" +
UMFMod.GetModVersion().ToString(), true);
            UMFGUI.RegisterPauseHandler(Pause);
            //MyFirstSRModConfig.Load();
        }
        public static void Pause(bool pause)
        {
            TimeDirector timeDirector = null;
            try
            {
                timeDirector =
SRSingleton<SceneContext>.Instance.TimeDirector;
            }
            catch { }
            if (!timeDirector) return;
            if (pause)
            {
                if (!timeDirector.HasPauser())
timeDirector.Pause();
            }
            else timeDirector.Unpause();
        }
        /*void Update()
        {
            if (!Levels.isSpecial() &&
SRSingleton<SceneContext>.Instance?.GameModel != null) //Makes
sure we are in game and that the GameModel exists.
            {
                playerModel =
SRSingleton<SceneContext>.Instance.GameModel.GetPlayerModel();
            if (playerModel != null) //Make sure that the
PlayerModel has been retrieved.
            {
                playerModel.maxHealth =
MyFirstSRModConfig.MaxHealth; //Set the max health to our config
value.
            }
        }*/
    }
}
```

- 2. Rename Patch_PURPOSEOFPATCH.cs to Patch_MaxHealth.cs along with the class name for it.
- 3. Add the following to the top of Patch_MaxHealth.cs:

using MonomiPark.SlimeRancher.DataModel;

4. Set **typeof** in the first HarmonyPatch attribute to use the **PlayerModel** class we discovered with dnSpy.

```
[HarmonyPatch(typeof(PlayerModel))]
```

5. Set the second **HarmonyPatch** attribute to the **ApplyUpgrade** function we discovered with dnSpy.

```
[HarmonyPatch("ApplyUpgrade")]
```

6. Inside the patch class create the following postfix function:

```
public static void Postfix(PlayerModel __instance)
{
}
```

7. Add the following line to our new function:

```
___instance.maxHealth = MyFirstSRModConfig.MaxHealth;
```

- At this point the max health will be set regardless of which health upgrades the player has due to this function being run everytime a save is loaded.
- Since this function is only run on a save load or when the player buys a new upgrade, changes to the config will not take effect immediately like this.
- 8. Your Patch_MaxHealth.cs should now look something like this:

```
using UnityEngine;
using HarmonyLib;
using MonomiPark.SlimeRancher.DataModel;
namespace MyFirstSRMod.Patches
{
    [HarmonyPatch(typeof(PlayerModel))]
    [HarmonyPatch("ApplyUpgrade")]
    static class Patch_MaxHealth
    {
        public static void Postfix(PlayerModel __instance)
        {
            __instance.maxHealth = MyFirstSRModConfig.MaxHealth;
        }
    }
}
```

- A Harmony **Postfix** patch will make the code in it execute at the end of the function when all other code in the original function has executed.
- You could change it to a **Prefix** by simply renaming the function to **Prefix**. This would cause the code to be run before the other code in the original function. However that would not work for this scenario since the other code then overwrites our max health again.
- 9. See Building for the next steps.

Method 3 (Harmony Transpiler)

This method will show you how you can use a Transpiler to overwrite code in memory rather than inject new code into an existing function.

This method is really solid for when you really need to modify something that can't be otherwise modified with Method 1 or 2.

Method 4

Coming tomorrow

Method 5

Coming tomorrow

Building

- 1. In Visual Studio in the top bar menu click **Build > Rebuild solution**.
- 2. Start the game and test if your mod works.
- 3. Proceed to test out the other methods or see Finalizing.
 - Don't forget to comment out the code from each method so it doesn't affect your test results.

Finalizing

Since this is just an example mod you should not publicize it anywhere since anyone can easily do this.

However for the purpose of completion these steps should be taken before releasing any mod.

- Edit ModInfo.txt to whatever you want to show users who install or update the mod.
- Edit Properties\AssemblyInfo.cs and fill in all the details of your mod.
- Edit configVersion in MyFirstSRModConfig.cs to match the version in AssemblyInfo.cs.
- Clean and remove or comment out any unused code.

From: https://umodframework.com/wiki/ - **UMF Wiki**

Permanent link: https://umodframework.com/wiki/guide_firstsrmod?rev=1562222225

Last update: 2019/07/04 07:37

