Tutorial: Updating old (pre-2.3.0) module mods to the new plugin system

Here's a quick guide on how to update an old module mod to work with the new BepInEx plugin system.

Note, that the BepInEx documentation does a better job at explaining the basics, but it does so assuming you're making a plugin from scratch. I'd still highly recommend reading the documentation to have a better understanding of how BepInEx plugins work. This guide is meant to be a quick reference for those that want to update their clientside mods and don't want to spend too much time on it. Link to BepInEx docs

1. Make sure your mod project targets `.NET Framework 4.7.2`
2. Add BepInEx as a dependency in one of two ways:
    1. Move `BepInEx.dll` into your mod project directory and add it to project references. You might need to also add `UnityEngine.dll` and `UnityEngine.CoreModule.dll` as references.
    2. Alternatively, add the BepInEx NuGet package source if you don't have it already (see footnote 1 for instructions), and then add a new NuGet package reference to `BepInEx.BaseLib 5.4.17`. This is the method I would recommend.
3. Update the mod entry point (usually Program.cs) as follows:
    1. Set the main class to inherit `BaseUnityPlugin`
    2. Add an attribute to the main class, called `BepInPlugin` and fill it with the necessary information (GUID, Name, Version)
    3. Move all the initialization code out of the `static void Main(string[] args)` method, and into either one of the following:
        1. The class constructor (will run the code as early as possible)
        2. The `void Awake()` method (will run the code as early as possible, but after the constructor)
        3. The `void Start()` method (will run the code once all the plugins are loaded)
4. Update any logging to utilize the new logger that's provided by the `BaseUnityPlugin` class
5. Don't forget that SPT ships with `ConfigurationManager` by default, so if there is anything that could be user-configurable in-game, be sure to utilize it extensively (see footnote 2 for extra information)

Examples

Code

```
//                                      Old                                      mod
namespace                                                                    YourMod
{

    {

        {



        }
    }
}
```

Display More

Code

```
//   Updated   mod   -   has   optional   ConfigEntry   lines   as   an   extra
namespace                                                                YourMod
{
                                                [BepInPlugin("com.username.modname",          "
                                        public         class         YourModPlugin         :
    {
//Thefollowingtwocodelinesarepurelyexamplesofutilizingthenewconfigfeatures
        // These are entirely optional and are not needed for your plugin to work
                                        internal    static    ConfigEntry<KeyboardShortcut>
                                                                  internal        static


        {



  //The lines below are part of the config definition and are entirely optional -
                                                                                //     they're     only

                "Keybinds",
                "ToggleSomething",

                                                                                "The    keyboard    sh



                "SliderOption",
                1f,

           "Some sort of setting that comes with a slider for configuring!",

        }
    }
}
```

Display More

Footnotes:

1. To add the BepInEx NuGet source, do the following (for Visual Studio 2017 and newer):
    1. Go to `Tools -> NuGet Package Manager -> Package Manager Settings`
    2. Go to `Package Sources`, and click the button to add a new source
    3. Set Name: `BepInEx`, Source: `https://nuget.bepinex.dev/v3/index.json`
    4. Click `Update`
2. For documentation on setting up a general config, check out the BepInEx docs section about it here. For advanced ConfigurationManager display options, like drop-downs and sliders, refer to the ConfigurationManager readme here.

Additional information

Programming skills required	1
Technical knowledge requirement	Intermediate

Prerequisites

Tutorial prerequisite list	Any IDE for working with .NET (Visual Studio 2022 recommended)