SDSCSMM v0.1-beta User Guide

Contents

1 Basic Usage

1.1 How to Use

The mod manager is relatively simple to use if you want to install mods. There are three main stages to mod installation:

1. Registering mods in the manager
2. Activating registered mods
3. Installing mods to the game archive

We'll go through each of these three steps in the following sections. First, take a moment to familiarise yourself with the annotated image of the user interface in figure 1.1. We will explore these sections as we go. For now, we'll just cover two sections:

- The "Game Executable Location" displays the location of the game on your system. You are prompted to set this the first time the mod manager loads. You can re-set this by clicking the "..." following the text box, if necessary. You should also be prompted to set the location if it is invalid, when the mod manager needs to access it.
- The "Message Log" will display time-stamped messages from the mod manager, including error messages if things go wrong. Sometimes this will notify you that a bug has occurred, and sometimes it will tell you that you have tried to do something wrong or that a mod is not set up correctly, but most of the time it should just inform you about what the mod manager is doing when the UI is locked.

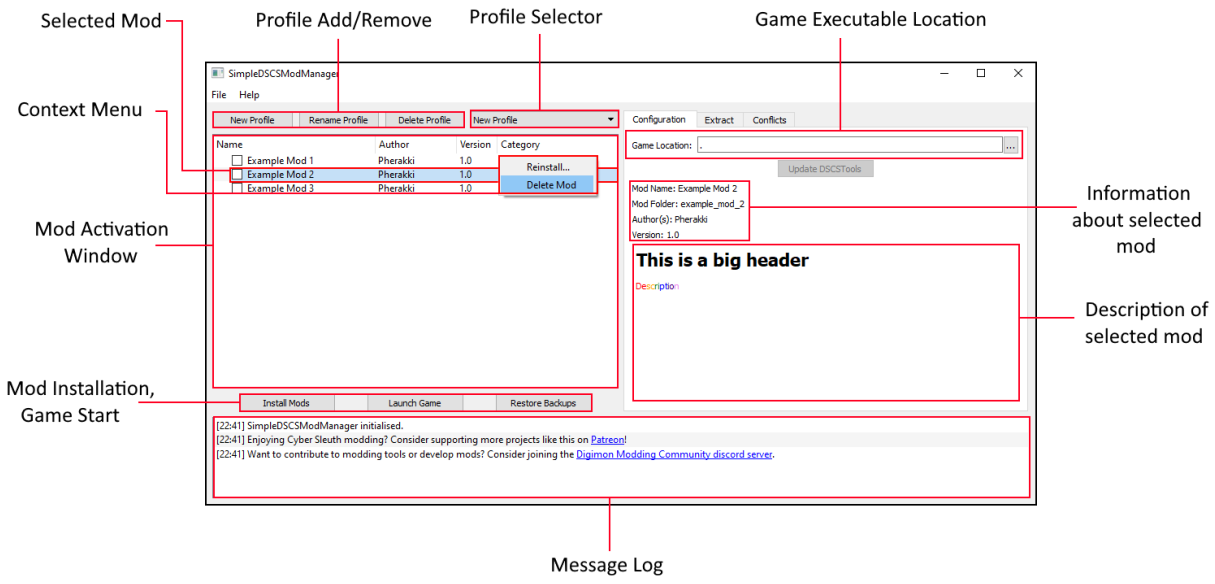

Figure 1: The mod manager user interface.

1.1.1 Registering Mods

Registering a mod with the manager "installs" the mod to the mod manager. This allows the manager to install the mod into the game files.
Registered mods are displayed in the **Mod Activation Window**.

There are two ways to register a mod with the manager. The first is to open **"File > Add Mod..."** in the toolbar, which will open a file dialog. A mod contained in a zip archive an then be selected from this file dialog.
The second and arguably simpler way is to drag-and-drop the mod into the **Mod Activation Window**. Any supported mod format can be installed this way, including an unarchived folder in the correct mod format. If the mod uses a CYMIS installer, you will be prompted to select what install options you want. It is possible to re-install such a mod with different options *via* the right-click **context menu**.

If you ever want to unregister a mod, you can delete the files from the mod manager *via* the right-click **context menu**.

1.1.2 Activating Mods with Profiles

This mod manager makes use of a **profile** system to allow different combinations of mods to be installed. You can switch between profiles by clicking the **profile selector**, which will display all the profiles you have created.
You can add, rename, and remove profiles by using the buttons to the left of the **profile selector**. Note that at least one profile must always be present, and the mod manager will prevent you from deleting the final profile.
To select which mods are to be installed by the **profile**, click the checkboxes next to the mod names. If it is checked, it will be installed during the next step.
If you click on a mod in the **Mod Activation Window**, then metadata about that mod will be displayed in the **"Configuration"** tab on the right-hand side of the manager. This will summarise the name of the mod, the name of the folder it is registered in in the manager files, the version, and category. Below these data, there is also a space to render a mod description in HTML.

1.1.3 Installing Mods

This is by far the simplest step for the user, but also the place where things are most likely to go wrong. Now that you have activated some mods, click the "Install Mods" button below the **Mod Activation Window**. This will begin the process of patching your selected mods together, and installing them into the game files. Depending on the number and size of the mods that are active, this can take several minutes to complete.
The adjacent button will launch the game, for convenience.
If you ever want to restore the game to a vanilla state, click the "Restore Backups" button, which is located to the right of the "Launch Game" button.

2 Advanced: Writing Mods, Getting Data, and Extensions

2.1 Writing a Compatible Mod

Mods require two things at a minimum:

- A folder called **modfiles**
- A file called **METADATA.json** containing a JSON dictionary

The "modfiles" folder contains the mod data. It should be structured like an unpacked game archive, with the "modfiles" folder in the place of the folder the unpacked game data would be in (i.e. "modfiles" might contain the folders "data/", "message/", "script64/" *etc*).
The **METADATA.json** can contain five main keys:

- Name
- Author
- Version
- Category
- Description

All of these should be strings (*i.e.* inside quotes), and additionally the "Description" field may be written in HTML.
There are two more keys that may also be used:

- Archives
- Rules

"Archives" is a dictionary that pairs each file contained in "modfiles" to the archive it should be installed into. By default, every file is installed to **DSDBP**, but if any files should be installed to different archives (e.g. **DSDBbgm**) then that can be set using this parameter.
"Rules" is a dictionary that pairs each file to the rule that should be used to patch that file together with other files of the same name from other mods. The default rule applied to a file depends on its type:

- MBE tables (csv files inside a folder ending with .mbe) are set to **mberecord_overwrite**
- Script source code (txt files inside script64/) are set to **squirrel_overwrite**
- All other files are set to **overwrite**

The currently available rules are:

- **overwrite** — replaces any file with the same name
- **mberecord_overwrite** — overwrites any lines in the target csv with the same IDs as the mod csv, creating new lines if no match exists
- **mberecord_join** — combines non-zero data in the target csv with non-zero data in the mod csv for lines with the same ID
- **squirrel_overwrite** — overwrites any functions in the target file with those in the mod file if they share a name, and adds any code outside of a function at the start of the file to the start of the target file

These rules allow you to just include the **new** data your mod provides, without needing to also include original game data that is irrelevant.

If your mod would benefit from some configuration options upon registry, *e.g.* there is an option of which textures to install, then your mod can also be set up to generate a customised set of "modfiles" with a **CYMIS document**. The details on how to write a CYMIS document are given in the accompanying text "cymis_specification". The CYMIS document must be saved as **"INSTALL.json"** alongside the "modfiles" and "METADATA.json". Note that a CYMIS-ready mod **should not have any files in "modfiles"**, since they will be wiped if the user ever re-installs the CYMIS-ready mod from within the mod manager.
Another use case of the CYMIS documents is to generate a large number of similar files from a single source file, which can substantially reduce the distribution size of a mod.

2.2 Extracting Data

Game data can be extracted by heading over to the **"Extract"** tab of the mod manager. The left-hand column of buttons will extract the game archive named on the button, including the unpacking of MBE database files and the decompilation of scripts. Clicking one of these buttons will open a folder selection dialog, where you can select which folder to extract the data to.
The right-hand column of buttons requires you to manually select an archive to pack or unpack. In addition, there are options to pack and unpack folders containing MBEs, and compile or decompile folders containing scripts.

2.3 Plugins

Several pieces of manager code are implemented as **plugins**. This allows key functions of the mod manager to be extended by dropping Python source files into the "plugins" directory. This feature is a work-in-progress and will be expanded upon in the future.