Modding Manual WARNO v1

Creating a new mod

Go to the game installation directory (in steam: Library -> WARNO -> Properties -> Local Files -> Browse Local Files), and enter the `Mods` subdirectory

Open a console in the `Mods` directory (either by using `cd <path_to_Mods_folder>` or by shift + right click in windows explorer > "open console here") and run `CreateNewMod.bat` with the mod name as argument. Example:

```
CreateNewMod.bat MyAwesomeMod
```

The name must be unique ; it will not work if a folder already exists with that name. If all goes well, you should see a message indicating that the mod was created, and a new folder named `MyAwesomeMod` has appeared. It should contain the following directories:

- AssetsModding
- CommonData
- GameData
- Gen

and the following files:

- base.zip
- GenerateMod.bat
- UpdateMod.bat
- UploadMod.bat

These 4 files (`base.zip` and the 3 `.bat`) are part of the modding tools, please do not modify them or you will likely compromise your mod.

If something is missing, try to delete the mod folder and recreate it.

You are now able to start working on your mod, please read on!

Generating a mod

A generation step is needed before a mod can be activated in-game or uploaded to the Steam Workshop. To proceed, launch **GenerateMod.bat** in your mod directory. The screen will become black for a short period of time.

Once you have generated your mod a default SteamID will be given to it. You need to upload your mod to give him a real SteamID. Even if you haven't finish your work, you can upload it and hide it on your workshop mod page.

Editing a mod configuration file

Go to the *C:\Users\USERNAME\Saved Games\EugenSystems\WARNO\mod*`MyAwesomeMod` folder, and open the Config.ini file:

```
Name = MyAwesomeMod ; The name of your mod
Description = My Mod is Awesome ; The description of your mod
TagList = ; Don't edit that field
PreviewImagePath = C:\my_mod_image.png ; An image located on your hard drive
CosmeticOnly = 0 ; Does your mod affect gameplay or not ? If activated, the
mod will not be shared when creating a lobby
ID = 902495510 ; Mod Steam ID, do not modify
Version = 0 ; Value to increment when an update in this mod is incompatible
with the current version
DeckFormatVersion = 0 ; Increment this value to invalidate all decks for this
mod
ModGenVersion = 0 ; Don't edit that field
```

If you want to change the mod name, you have to : change the "Name" field in the configuration file, change the folder name in *C:\Users\USERNAME\Saved Games\EugenSystems\WARNO\mod*, change the folder name in the steam directory.

The "ID" and "TagList" are filled when UploadMod.bat is launched.

Uploading a mod

To upload your mod on steam, launch **UploadMod.bat** in your mod directory. The screen will become black for a short period of time.

This process will update "TagList", "ID" and "ModGenVersion" fields.

Once you've uploaded your mod, it's ready to use on steam workshop, you can access its own page, set its visibility, description, image, etc...

Updating a mod – When a new game patch is released

With the game updating through Steam, it will be needed from time to time to update your mod so that it still work with new versions. We created a small tool to help you with this, requiring minimal manual work. Before talking about it, let's see what the problem is.

Why do we need to update mods?

Imagine you have a mod that modifies file `A.ndf` . When this mod was created, the original file (from game data) was:

```
Thing is TThing
(
        SomeInt = 42
)
```

And you modified it to be:

```
Thing is TThing
(
        SomeInt = 43
)
```

No problem here, that's the point of modding. Now what if we at Eugen need to modify this file as well, and end up adding content to it so it looks like this in the new version of the game:

```
Thing is TThing
(
        SomeInt = 42
)

Thing2 is TThing
(
        SomeInt = 80
)
```

You will want to keep your change to `Thing/SomeInt`'s value, and have the new `Thing2` as well. Wanted result for your mod:

```
Thing is TThing
(
        SomeInt = 43
)

Thing2 is TThing
(
        SomeInt = 80
```

```
)
```

Updating mods

The solution is quite easy, just run `UpdateMod.bat` in your mod folder. This will merge your changes with the last version of the game data.

If you're lucky, the process will work automatically and display a message indicating success. However, it is possible that you encounter *conflicts* during the operation. Conflicts arise when both the new version of the game and your mod modify the same part of a NDF file. In this case, the process can't guess what result your want, and need manual intervention.

Solving conflicts

Imagine once again that you're modifying `A.ndf`.

Original version (game data when the mod was created):

```
Thing is TThing
(
        SomeInt = 42
)
```

Modded version:

```
Thing is TThing
(
        SomeInt = 43
)
```

And with the following update, the game's version of the file becomes this:

```
Thing is TThing
(
        SomeInt = 80
)
```

Both the game and the mod have modified the same line. You can easily see that it is entirely up to you what final value should `Thing/SomeInt` have in your mod, and that no tool can decide it on its own.

The updating tool will stop and warn you about the files that contain conflicts. To handle this case, the tool will mark conflicts like this one with special delimiters:

```
Thing is TThing
(
<<<<<<<
        SomeInt = 80
```

```
|||||||
        SomeInt = 42
=======
        SomeInt = 43
>>>>>>>
)
```

- After `<<<<<<<` is the new version from the game's data
- After `|||||||` is the base common ancestor (in our case original game's data)
- After `=======` is your mod's version
- `>>>>>>>` marks the end of the conflict

Resolving the conflicts consists in modifying the file so that it makes sense to the game that will load it (ie. get rid of the delimiters and have a single `SomeInt = ...` line) and has the wanted value for your mod. You are free to keep your mod value (43), to go back to the original value (42), to get the new value (80) or to decide an entirely new.

The process is very similar to a three-way merge like found in version control software such as git.

Mods backup

Inside your mod directory you will find 2 essentials scripts :

- CreateModBackup.bat : which will allow you to create a backup of the current status of your mod. Backups are basically a copy of your mod files archived and stored in `Backup/` folder. A backup will be generated every time you upload your mod, considering that if you upload your mod that means that you're currently on a stable version. You can generate a backup yourself by using this script it will ask for a name, you can leave the entry empty it will use a generated name based on current time.
- RetrieveModBackup.bat : this script will allow you to reset your mod to a previous version stored in `Backup/`, it will simply copy and force paste the backup version in your current working space so be careful before any RetrieveModBackup. The script will prompt you for a name, you can leave the entry empty it will select the lastest backup generated.

NDF Map

Every files location you could want to edit.

Interface

If you want to mod the interface, you gonna need to look at `GameData/UserInterface/`.

All files under `Style/` folder will be related to global objects and reused style accross different screens.

All files under `Use/` will describe each screen or part of screen, separated in folder following their usage in game :

- InGame : everything after a match session is launch
- OutGame : everything related to main menu, loading, etc...
- Common : everything common to both In and OutGame.
- ShowRoom : well it speaks for itself, it's everything related to the showRoom.
- Strategic : everything related to Steelman mode

Gameplay

Units

All unit descriptors are located in `GameData/Generated/Gameplay/Gfx/UniteDescriptor.ndf`. You will find various other interesting descriptors in the same folder, like `WeaponDescriptor.ndf` and `Ammunition.ndf`.

Divisions

Divisions are located in `GameData/Generated/Gameplay/Decks/Divisions.ndf`. Divisions are composed by packs, a pack is a bundle of the same units, all packs are located in `GameData/Generated/Gameplay/Decks/Packs.ndf`.

Gameplay constants

`GameData/Gameplay/Constantes/` contains all files defining generic gameplay features, for example:

- in `GDConstantes.ndf` you will find the property DefaultGhostColors which define the color for ghosts
- RelativeBonusMoneyByIADifficulty allow you to give more income money to the AI.

Weapons

Weapons are mainly described in `WeaponDescriptor.ndf` and `Ammunition.ndf`, located in `GameData/Generated/Gameplay/Gfx`.

The first file describe, for each units, how its turrets are configured, and which weapons are mounted. For example, you'll find the properties defining the number of salvos for each weapon, the angle ranges and rotation speeds of the turrets, and some UI-related parameters. You will also find reference to `TAmmunitionDescriptor` objects, which are defined in `Ammunition.ndf`.

In these objects there are many useful properties, here are some of them:

- `Arme`: the type of damage the weapon will inflict. For AP weapon, you can use values from `AP_1` to `AP_30` (see `CommonData/Gameplay/Constantes/Enumerations/ArmeType.ndf`).
- `PorteeMaximale`: range of the weapon. Some of you already figured out how the meaning of the unit (cf. this post).
- `HitRollRuleDescriptor`: describes how the hit and pierce chance are computed (see `GameData/Gameplay/Constantes/HitRollConstants.ndf`).
- `IsHarmlessForAllies`: enables/disables friendly fire.

There are also other files that you might want to take a look at:

- `GameData/Gameplay/Unit/CriticalModules/CriticalEffectModule_GroundUnit.ndf`: describes the critical effects distribution on ground units (typically tanks).

AI

`GameData/Gameplay/Skirmish` contains files about the AI. An artificial intelligence is called strategy.
There are multiple strategies (used in skirmish, multiplayer, historical battles and Steelman) depending on game difficulty and settings, located in *GameData/Gameplay/Skirmish/Strategies/* subfolders.

Strategies are plugged in the `GameData/Gameplay/Skirmish/IASkirmishDescriptors.ndf` file.

A strategy contains a list of Generators and Transitions. The AI navigates from one TSequenceGeneratorDescriptor to another depending on TIAGeneralStrategyTransition.

A TSequenceGeneratorDescriptor describes the AI's priorities while it is active. MacroAction behavioral descriptors used in the Generators are located in `GameData/Gameplay/Skirmish/SkirmishMacros` folder. Most MacroActions control unit groups with specific objectives (attack, defend, etc.). The required units for each action are described in TPoolModel objects which are freely modifiable. Note: new TPoolModel objects need distinct GUID fields.

Global settings about the AI are defined in `GameData/Gameplay/Constantes/IAStratConstantes.ndf` and affect both skirmish and campaign AI.

Creating your first mod, step-by-step

This section objective is to create a mod which allows to create decks with more units (100 slots points instead of 50).

Follow instructions of the first section of this manual, called **`Creating a new mod,`** to create your mod MyAwesomeMod.

From your mod folder, open file `GameData/Generated/Gameplay/Decks/Divisions.ndf.`

Replace all `MaxActivationPoints = 50` to `MaxActivationPoints = 100.`

Follow instructions of section **`Generating a mod`** to generate your mod.

You can now activate and test your mod, in the game through the menu `Mod Center`.

Follow instructions of section **`Uploading a mod`**.

Congratulation, you have created and published your first mod in WARNO workshop.