STEEL DIVISION

NORMANDY 44

MODDING MANUAL

EUGEN SYSTEMS

paradox INTERACTIVE

UPDATING A PREVIOUS MOD (V.77875)

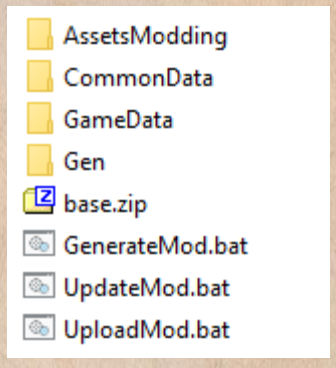
This procedure is mandatory if you wish to keep your mod working well on new versions of the game. This will be needed only once, after that the process will be automated.

This section only concerns mods created on the first version of modding, game version 77875.

How to be sure that your mod needs this procedure: open your mod directory (in steam: Library -> Steel Division -> Properties -> Local Files -> Browse Local Files, then choose your mod directory) and check that there is `no base.zip` file in it.

1. Copy the file `Mods/Utils/Legacy/base_77875.zip` into your mod directory, and rename it `base.zip`.
2. Copy the file `Mods/ModData/UpdateMod.bat` into your mod directory.

It should look like this:



Then run `UpdateMod.bat`. This will launch the usual procedure for updating mod, described further in this manual.

CREATING A NEW MOD

Go to the game installation directory (in steam: Library -> Steel Division -> 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 it a real SteamID. Even if you haven't finished your work, you can upload it and hide it on your workshop mod page.

In case you have generated 2 mods and uploaded none of them, they will both have the same default SteamID. Take care of uploading at least one of them to give it a real ID. Otherwise some features could be broken : the mod center will activate only one of them and always the first one in alphabetic order.

EDITING A MOD CONFIGURATION FILE

Open the `Config.ini` file in the next folder > `C:\Users\USERNAME\Saved Games\EugenSystems\SteelDivision\mod\MyAwesomeMod`

```
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's name, you have to : change the "Name" field in the configuration file, change the folder name in `C:\Users\USERNAME\Saved Games\EugenSystems\SteelDivision\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.

UPDATING A MOD WHEN A NEW 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 works 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 you want, and needs 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 choose an entirely new one.
The process is very similar to a three-way merge like found in version control software such as git.

NDF MAP

Every files' location you could want to edit.

INTERFACE

If you want to mod the interface, you're going to need to look at `GameData/UserInterface/`. All files under `Style/` folder will be related to global objects and reused style across 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 : old, unused, relative to strategic mode.

Adding a new texture

If you want to add a new texture to replace one already used in game, you can change its declaration by adding 'Modded' at the end of its type (`TUIResourceTexture_InGame` will become `TUIResourceTexture_InGameModded`, `TUIResourceTexture_Outgame` will become `TUIResourceTexture_OutGameModded`, etc.), then set its path relative from `AssetsModding/`.

```
All your interface assets must be located under AssetsModding/.
```

Localisation

You will sometime find variables called TextToken or NameToken. Every time you see a name containing "Token", it means that the string is supposed to be localised and the content of the variable is a reference token for localisation files.

You can find the mod localisation file here :
`GameData/Localisation/MODDING.csv`

You can either create your own token to replace some of our tokens or you can override our tokens' value. All you have to do is set the token value under the **TOKEN** column and set its content under **REFTEXT** column.

> Your token size cannot exceed ten characters!

GAMEPLAY

Units

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

Divisions

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

Gameplay constants

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

- in `GDConstantes.ndf` you will find the property `GhostDefaultColor` which define the color for ghosts
- `RelativeBonusFluxByIADifficultyAndPhase` and `RelativeBonusPhaseStartByIADifficulty` allow you to give more income money to the AI.

Weapons

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

The first file describes, 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 `GameData/Gameplay/Constantes/Enumerations/ArmeType.ndf`).
- `PorteeMaximale`: range of the weapon. Some of you already figured out the meaning of the unit (cf. this post).
- `HitRollRuleDescriptor`: describes how the hit and pierce chances 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).
- `GameData/Gameplay/Gfx/Generated/ArmesArmures.ndf`: the damage factor used given the firing weapon and the target's armor.

AI

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

- Skirmish and multiplayer strategy files: `GameData/Gameplay/Skirmish/Strategies/`
- Campaign strategy files: `GameData/Gameplay/Skirmish/Campaign`

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.

SOUND

You can add your own sound to SteelDivision : Normandy 44. To do so, you need to put them under `GameData/Assets/SoundModding/`. (Create folders if they are not already here)

To redirect an existing sound to your own sound, you need to replace its TSoundStream, Template_GlobalAcknow, Template_UnitAcknow or TemplateSoundDescriptor by the modded version :
TSoundStream -> SoundStreamModded
Template_GlobalAcknow -> Template_GlobalAcknowModded
Template_UnitAcknow -> Template_UnitAcknowModded
TemplateSoundDescriptor -> TemplateSoundDescriptorModded

Then set the FileName to your sound (the path is relative to `GameData/Assets/SoundModding/`. Example : you have your `GameData/Assets/SoundModding/MG_shot.wav` then your FileName will be "MG_shot.wav")

SPECS

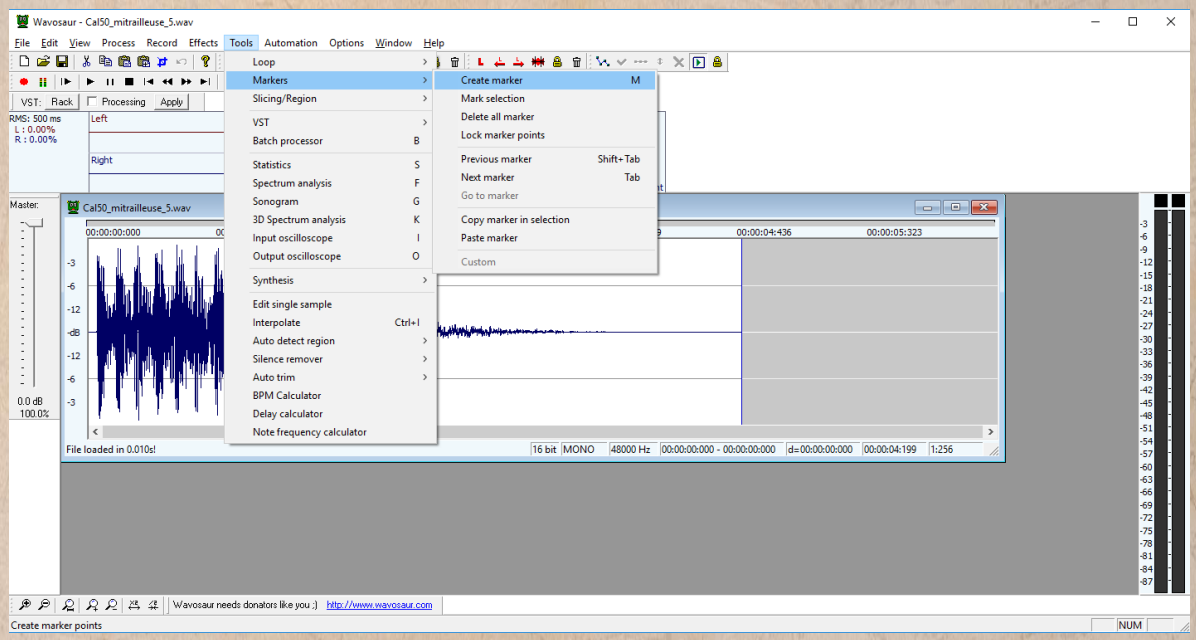
```
Sound Format > Only OGG and WAV files are supported!
```

Sound FX (shot fired sound, impacts, etc...) must be in a specific format : 16 bits MONO WAV (44100Hz).

Specific case: Continuous fire

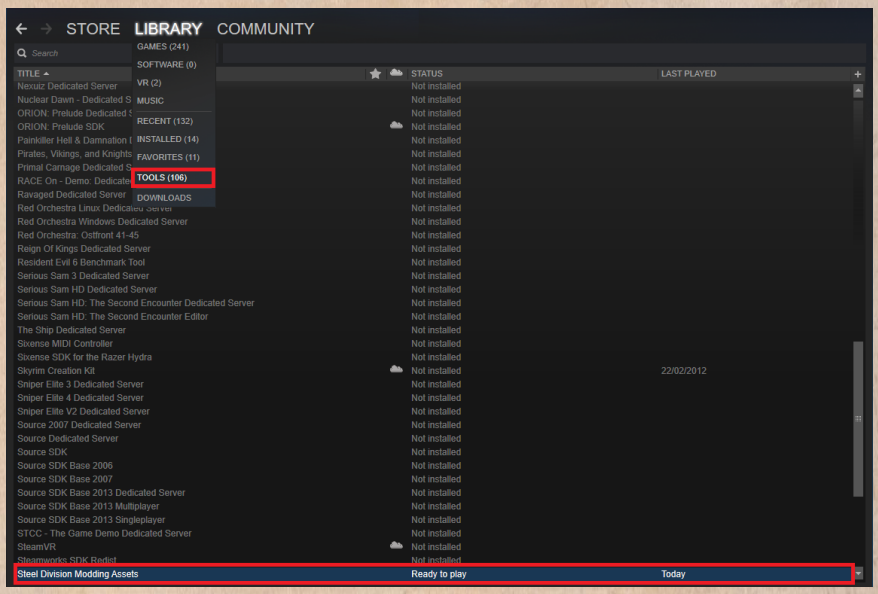
For continuous firing weapons you will need to mark a loop into your file. You need to getthe software Wavosaur, load you file then tools>Markers>Create a marker (shortcut 'M').

You need to mark the loop begin and end. That way, the sound will begin at the start of your file, then loop between the 2 markers until it ends, then it will play the end of the sound.



SKINNING

In order to reskin our units you will need to download a new Modding tool located on steam under Library > Tools > Steel Division Modding Assets



By downloading this tool you will be able to access our models textures and Ase2NdfBin (necessary to generate your mod). You will find them here : `steamapps/common/Steel Division Modding Assets/Mods/Assets/3D/Units`.

Each type of units is archived, you will need to unarchived it before selecting which unit you want to reskin. Once you have selected your units to reskin, you will need to copy the unit's files (all png or jpg and Ase2NdfBin) and past them in your mod, with the same path.

Example : if you want to reskin the US M4A1 Sherman, go to `steamapps/common/Steel Division Modding Assets/Mods/Assets/3D/Units/USA/Char.zip`, unzip it, go to `Char\M4A1_Sherman\`, copy all files and past them in `steamapps/common/Steel Division/Mods/[YourMod]/GameData/Assets/3D/Units/USA/Char/M4A1_Sherman/`

```
Warning LODs: Don't forget to copy and modify unit's LODs and
MID versions, otherwise you will get some surprise in-game.
```

Once you have copied and modify the unit's textures you can generate your mod and appreciate your work in game.

MESH MODDING

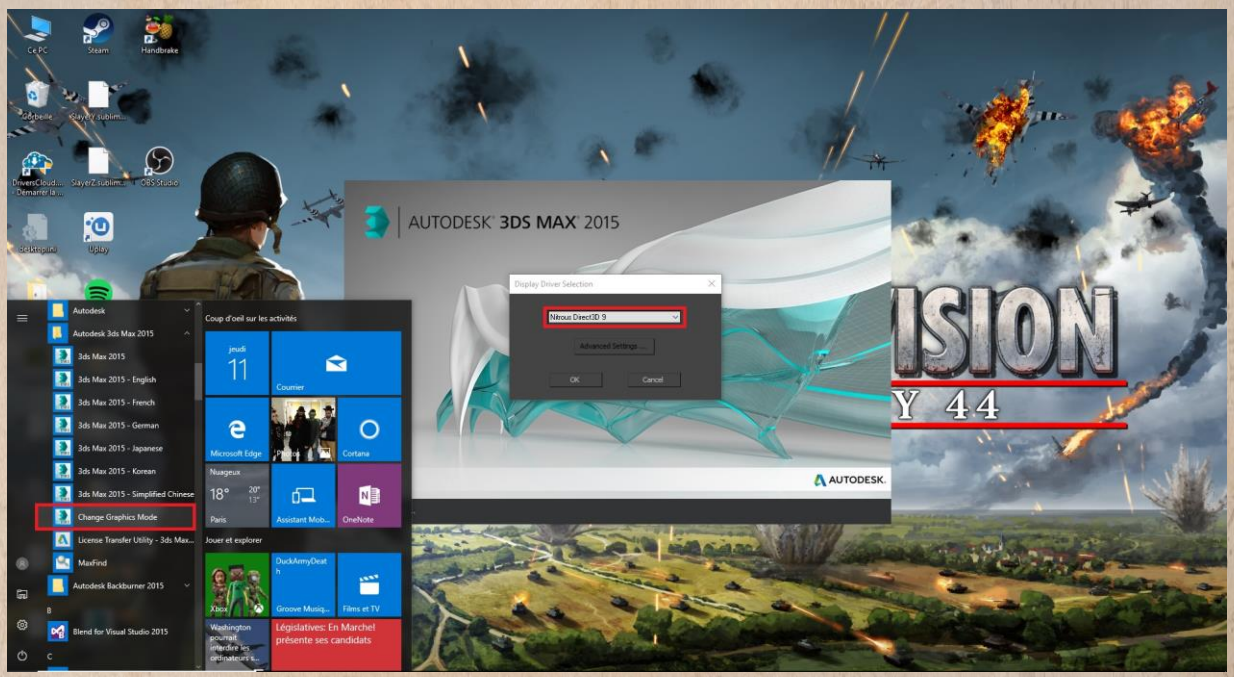
SET UP YOUR ENVIRONMENT

To create or modify mesh in order to integrate them in Steel Division: Normandy 44 you will need 2 plugins :

- `Ase2NdfBinPlugin_Install.exe`
- `eugMaterial_Install.exe`

Quit 3ds Max if it's running, install those plugins, you can find them under `steamapps/common/SteelDivision/Mods/Utils/Meshes/PluginsInstalers`, then restart 3ds Max.

Once you have done that you will need to setup your viewport settings. To do so, use Autodesk 3dsMax 2015/Change Graphics Mode and set it to "Nitrous Direct3D 9"

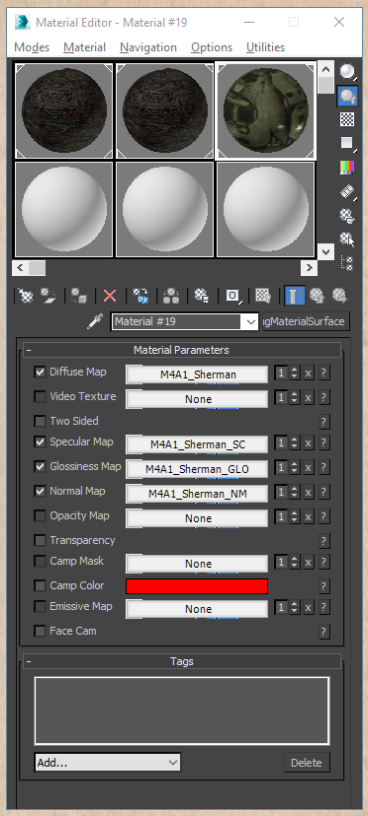


Ase2NdfBin

The file needs to be exported from 3dsMax in this format to be used ingame.

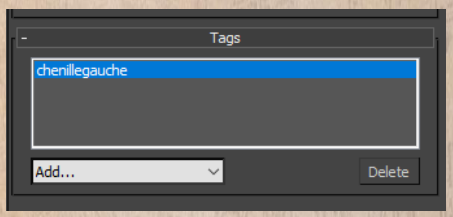
EugMaterial

EugMaterial = in 3dsMax, we don't use Standard Material. We use this specific material that simulates the ingame visual of 3d models.



Along with classic channels (Diffuse, Specular, Glossiness, Normal Map and Opacity), you'll notice a "Tags" section.

You can add tags on some elements to have specific behaviors. For example, by adding 2 different materials (with its own separated textures) with tags "chenillegauche" and "chenilledroite" on left track and right track of a tank, you'll see the tracks move ingame when the units is moving.



3D MODEL DESCRIPTION

To work properly ingame, a unit's 3D model needs different elements with a specific nomenclature (some words are in French).

Let's take the example of the M4A1 Sherman tank.

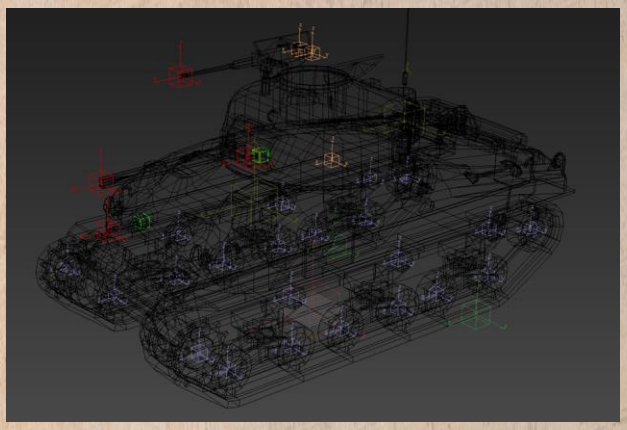
3D MODEL DESCRIPTION

To work properly ingame, a unit's 3D model needs different elements with a specific nomenclature (some words are in French).

Let's take the example of the M4A1 Sherman tank.

If you check inside the 3d file, you can see 2 sorts of elements that compose the model :

- **3d meshes parts : the guns, the turret, wheels, tracks, etc. These elements are visible ingame.**
- **"Point helper" boxes : These boxes are invisible ingame. We use the position (but also scale and orientation) of these boxes for different things like FX spawning points or procedural animations on some visible elements.**



3D MODEL SPECS

3D Mesh

LODs are not generated, so you need to make you own version of High, mid and low poly:

High = 6/7000 triangles max.
Mid = around 2000 triangles.
Low = less than 500 triangles.

Textures

High = 2048 x 2048 px Diffuse, Glossiness (_GLO), Specular (_SC), Normal Map (NM) and sometimes Alpha (_A) + 256 x 256 px separated textures for tank's tracks. This texture must be separated because when the unit is moving, UVs coordinates of tracks meshes will move too, to give the illusion of real tracks.

Mid = 1024 x 1024 px textures. This time, tracks textures must be inside the main textures. They don't move at this level of details.

Low = 256 x 256 px textures.

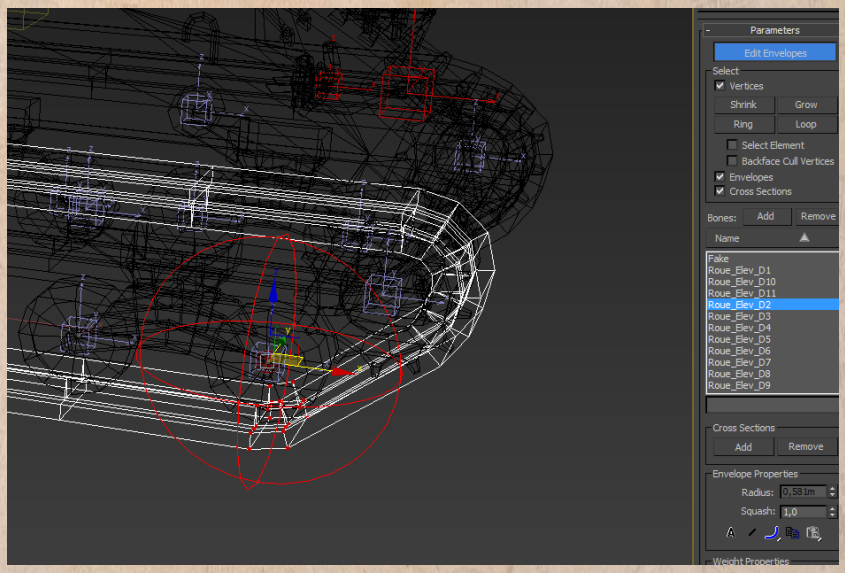
DIFERENT USES OF HELPERS BOXES

Fx helpers

- FX_Tourelle_x_tir_0x = When unit is firing, weapon FX will spawn here.
- FX_Stress_01/02 = Spawn position of smoke FX when unit is taking damages.
- FX_Fumee_Chenille_D1/G1 = Smoke FX when unit is moving.

Skinning helpers

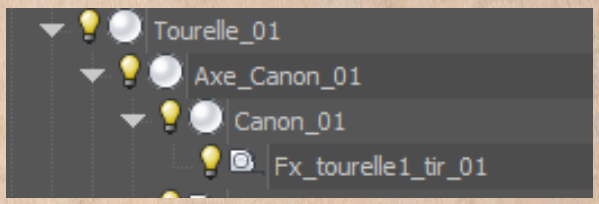
- Roue_Elev_Dx/Gx = Wheels meshes (Roue_Dx/Gx) are linked to this type of boxes. When the unit is moving, this box moves vertically and pulls up/down the wheel and also some vertices of the tracks, thanks to the "Skin Modifier". The rotation is applied directly on the wheel mesh.
- Fake helper = All tracks vertices not "skinned" to a wheel are attached to this box. This box is linked to the main part "Chassis". When the unit will move/stop, it will drag those vertices and we'll have a more realistic movement.



3D MODEL HIERARCHY CONSTRAINTS

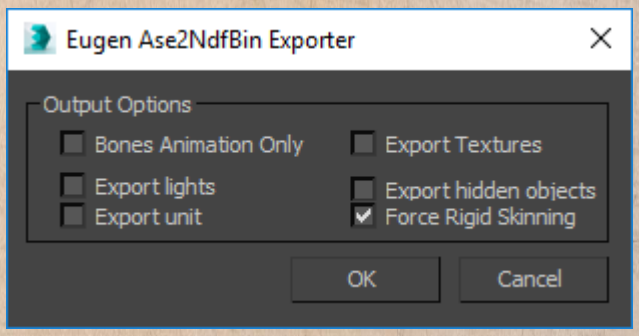
"Chassis" is the main element, it's linked to nothing, but lot of other elements are linked to it.

A "Turret" works always this way (-> means "linked to") : Fx_Tourelle1_tir_01 -> Canon_01 -> Axe_Canon_01 -> Tourelle_01.
Then the "Tourelle_01" element is linked to the chassis (or to another turret if it's a secondary turret).
Helpers (smoke, stress) can be linked to the chassis or to a turret, or to nothing.



EXPORT

As told earlier, once the model is ready, you must export in ".Ase2NdfBin" format.
The exporter will ask you what you want to export, you just have to check the box "Force Rigid Skining".



ADDING YOUR MESH TO YOUR MOD

Once you have your `Ase2NdfBin`, you have to move it with its textures under `GameData/Assets/` (you may have to create the path yourself). Otherwise your mesh will not be generated and will not be usable in the game.

Now you can add references to your mesh. For example you can replace the US Sherman M4A1 by modifying the unit mesh description located under `GameData/Gameplay/Gfx/Units/US/Char/Sherman_M4A1_US.ndf`.

You have to change all `FileName` to redirect to your own mesh and replace the `CivMask` variable by "`CivMask/MeshModding`".

COMMON ISSUES

- In case you have scaled your mesh, you need to run a ResetXForm after that.
- Some 3ds Max Modifiers can cause some errors during export, to be sure collapse as many Modifiers as possible before the export.