Adding a wholly custom new arena fight - from A to Z

by lugia19

Things you will need to do:

1. ArenaParam editing and Text editing (to add the fight itself)
   1.1) Editing ArenaParam, to add the new entry
   1.2) Adding the tab text for the new tab
   1.3) Adding a description for the fight
2. Character changes:
   2.1) Adding a new AccountParam entry and text, to set the AC/pilot name, and decal/emblem.
   2.2) Adding a new NpcParam
   2.3) Adding a new NpcThinkParam (To change which AI it uses, even to a custom one)
   2.4) Adding a new CharaInitParam (To reference a new design file)
   2.5) Editing/repacking param/asmparam/asmparam.designbnd.dcx (to add the new .design file)

   Everything after this step is technically optional - you can just add a fight with steps 1 and 2. You'll be able to have a custom build with a new design file, and you'll just have to re-use one of the existing AIs.
3. Adding images (except for the rank icon):
   3.1) Editing/repacking menu/hi/01_common.tpf.dcx and menu/hi/01_common.sblytbnd.dcx (to add a new decal to SB_DecalThumbnails, used for the icon next to the HP bar)
   3.2) Editing/repacking menu/hi/00_solo.tpfbdt (to add the Preview image and the Emblem)
4. Creating and adding a custom AI script for the fight:
   4.1) Copying an existing script and modifying the witchybnd xml
   4.2) Decompiling and modifying the LUA script to change the logicID
5. Adding new rank icons:
   5.1) Creating an image containing all the icons
   5.2) Generating a .layout file for your image
   5.3) Adding your rank icons to the game
6. Adding custom intro/outro dialog
   6.1) Adding entries to TalkParam
   6.2) Adding text to TalkMSG
   6.3) Adding audio

Let's break it down and go step by step.

Step 1 - ArenaParam and text editing

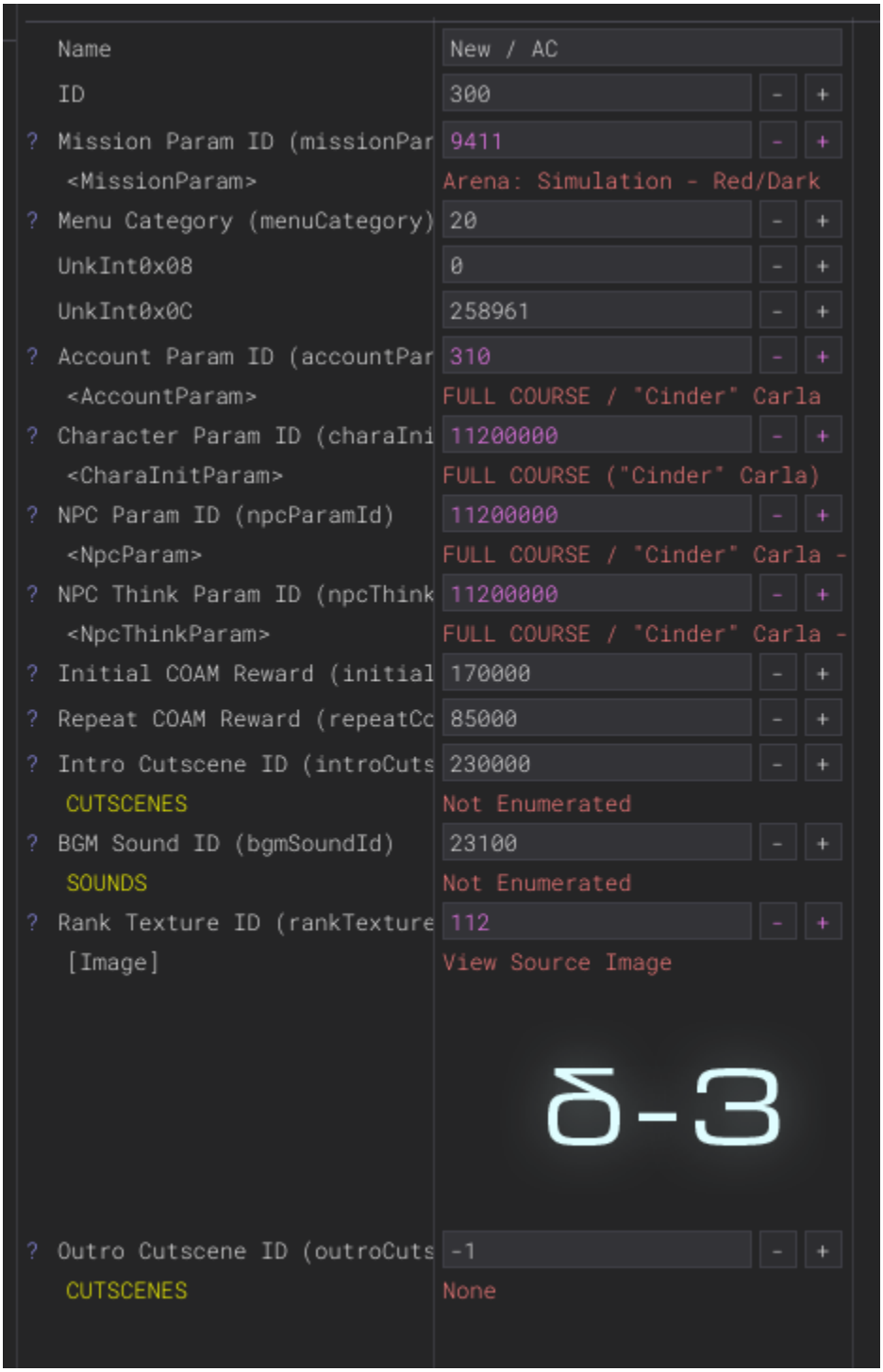
ALTERNATIVE: If you need to add MANY entries, you might want to check out this script I made.
If you export all params as CSVs and all the text entries from Smithbox, you can use it to add however many entries you want. Be sure to modify the parameters at the top of the file to make sure they match up with, say, your icons etc, but the defaults should work. This will also add the necessary params/text for the intro/outro.

1.1) Add a new entry in ArenaParam, by duplicating an existing one.

The values I've changed here are:

- ID: The existing convention is that each new page starts at a different set of hundred (page 0 starts at 100, page 1 starts at 200) so since we're adding a third page, we start at 300. Following entries would be 301, 302, etc.
- Menu category: This determines the tab. Setting it to 20 means we're creating a 3rd tab.

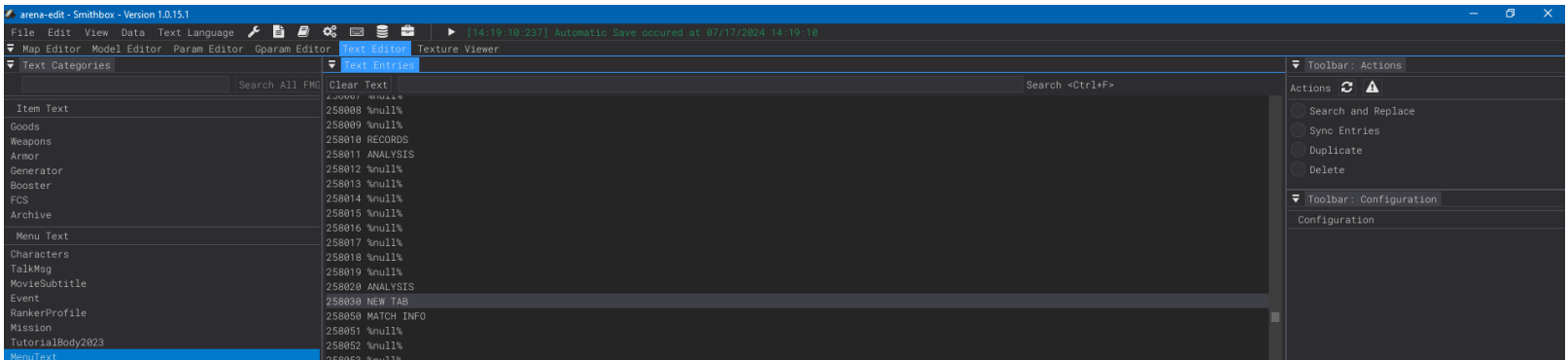
- MissionParam: Just determines the map.
- Rank Texture ID: Just determines the rank displayed. You'll see in step 5 how to add new ones.

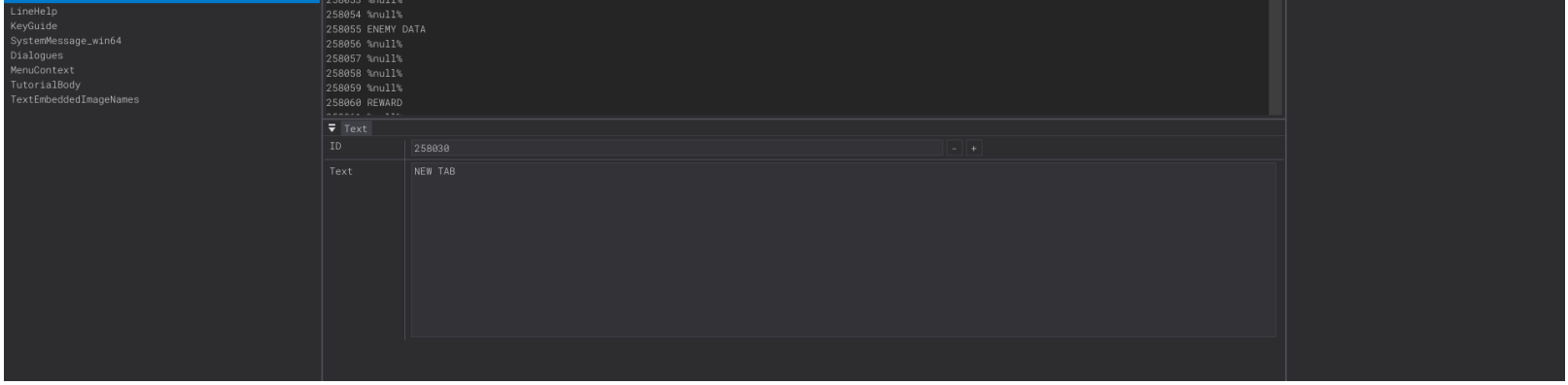


1.2) Giving the new tab a name.

The way the tab names are fetched is that the game looks for an entry with ID [258010+menuCategory] in MenuText.

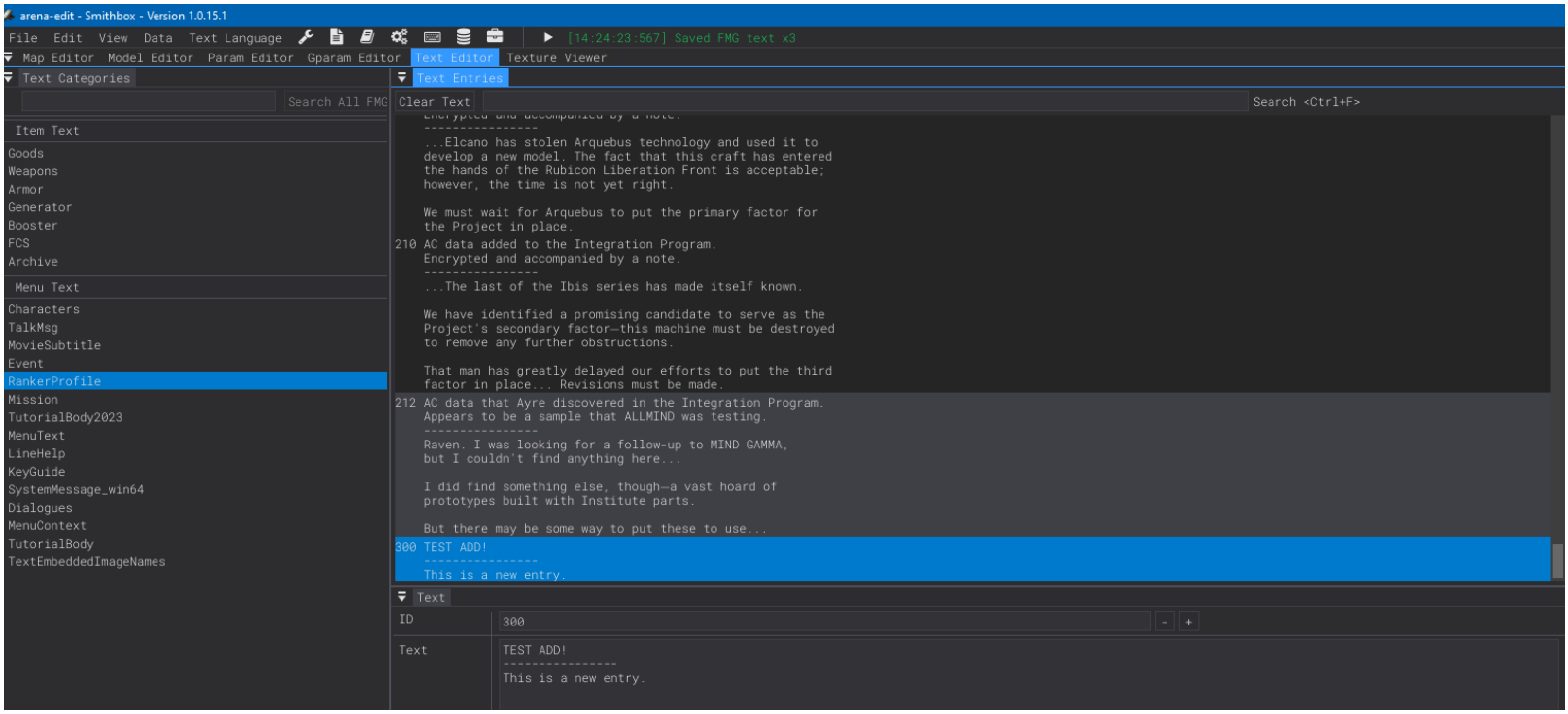
So we go in the Text Editor, and add a new entry with ID 258010+20 = 258030, containing our tab text.



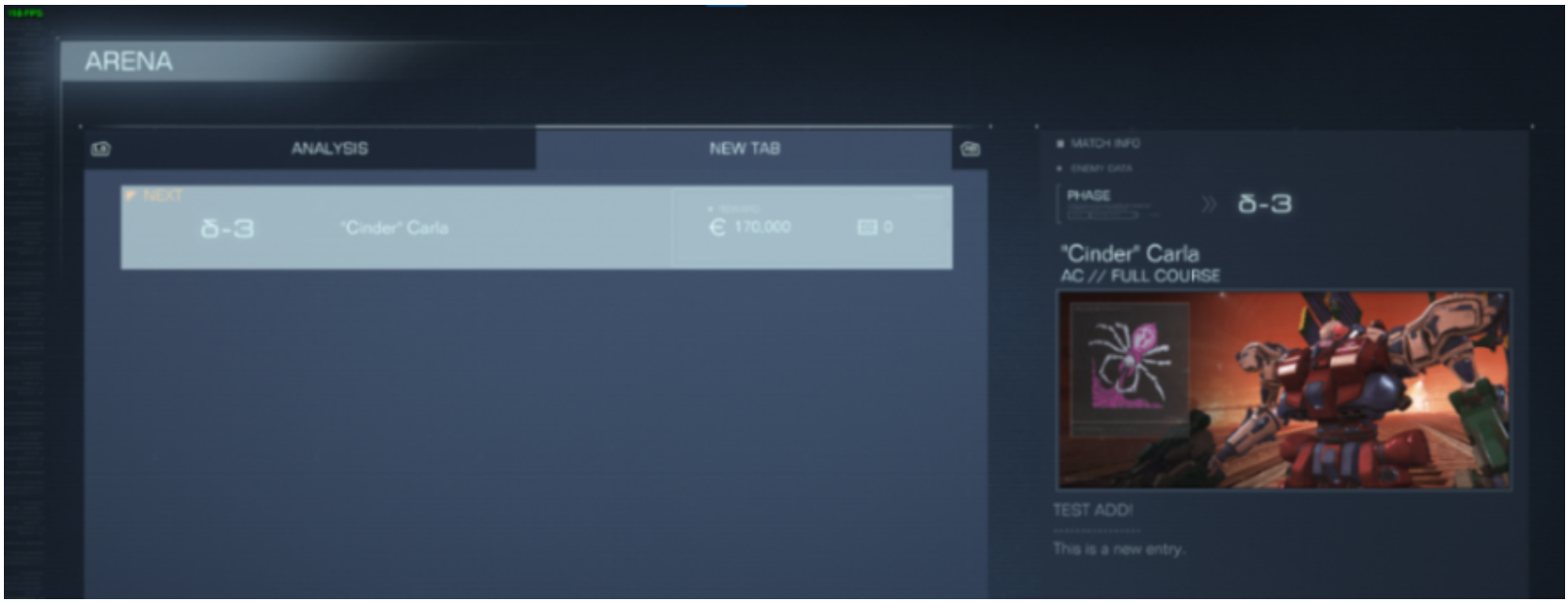


1.3) Giving the new entry a description

Very simple - in the text editor, under RankerProfile, the entries are stored with their ID. So we just make a new entry with ID 300.



Our work so far - a copy of an existing fight, in a new tab, with a new description (but as you can see, the name is still the same as the original):



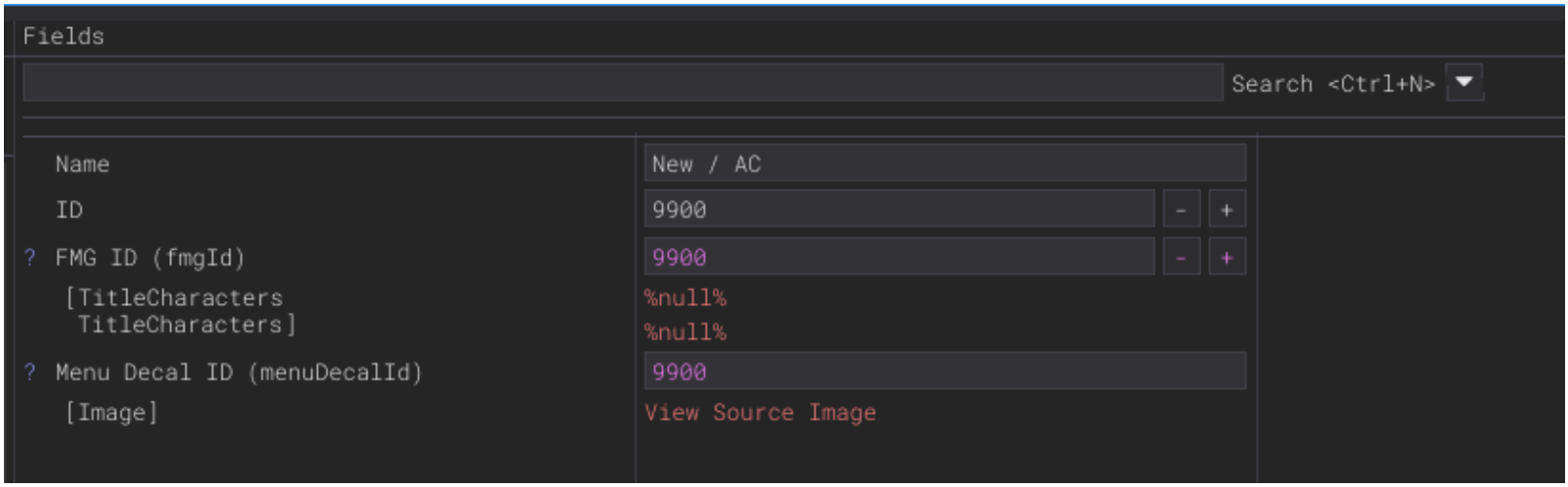
Step 2 - All the character stuff

In the new ArenaParam entry, we now have four entries we will be focusing on. These are:

- **AccountParamID**: Determines the AC/Pilot name, as well as the Decal image next to their health bar and the Emblem that appears during the intro.
- **CharaInitParamID**: Determines the actual AC's build, by referencing a design file.
- **NpcParamID**: Determines... No idea. But it also references an AccountParamID, so we'll have to change it accordingly here too.
- **NpcThinkParamID**: Determines which logic file/AI is used for this AC.

2.1) Giving our AC a new name by adding an AccountParam entry.

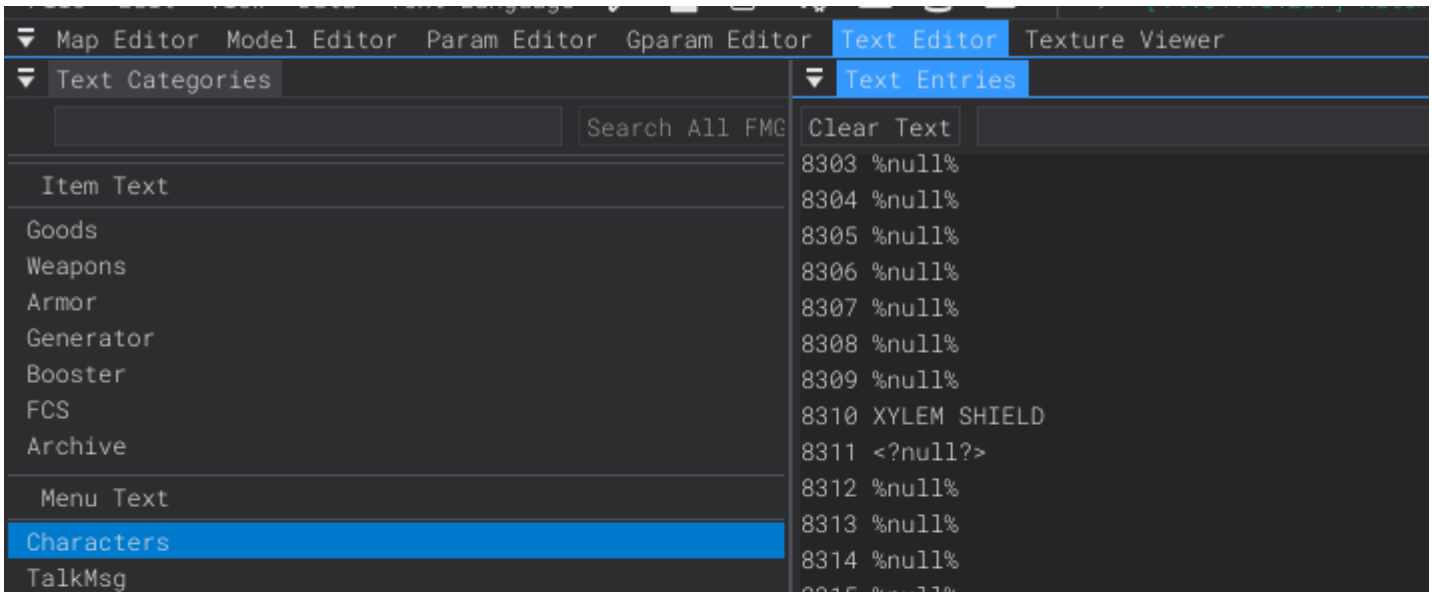
We duplicate an existing AccountParam entry. We can give it any unused ID we like. In this case, I'll do 9900.
I'll use that number for most things. We will also assign it a menuDecalId of 9900, even if it doesn't exist yet. It'll be added in a later step, for now it'll just display as empty.

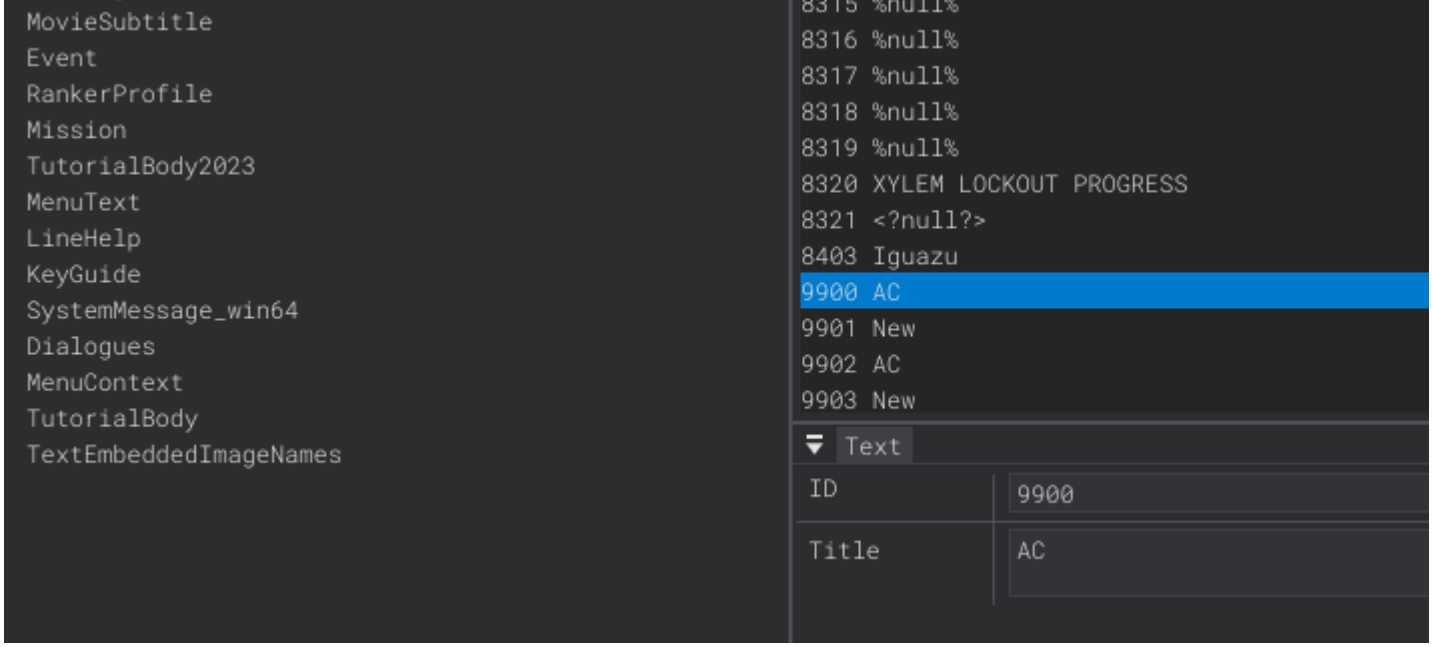


Now that we've done that, we need to actually add the AC and pilot name with the text editor.
The way they're looked up is simple - the lookup starts at the fmgId indicated in the AccountParam, and it expects to see, in order (yes, they're repeated twice for some reason):

- AC Name
- Pilot Name
- AC Name
- Pilot Name

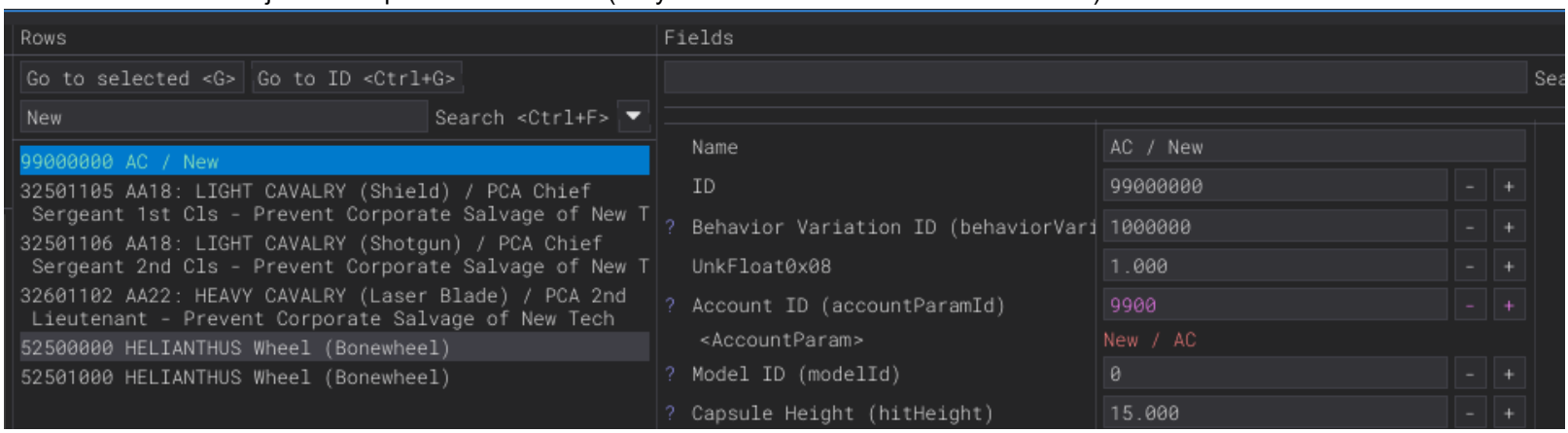
So in our case, we'll add new entries starting at ID 9900.





2.2) Creating a new NpcParam.

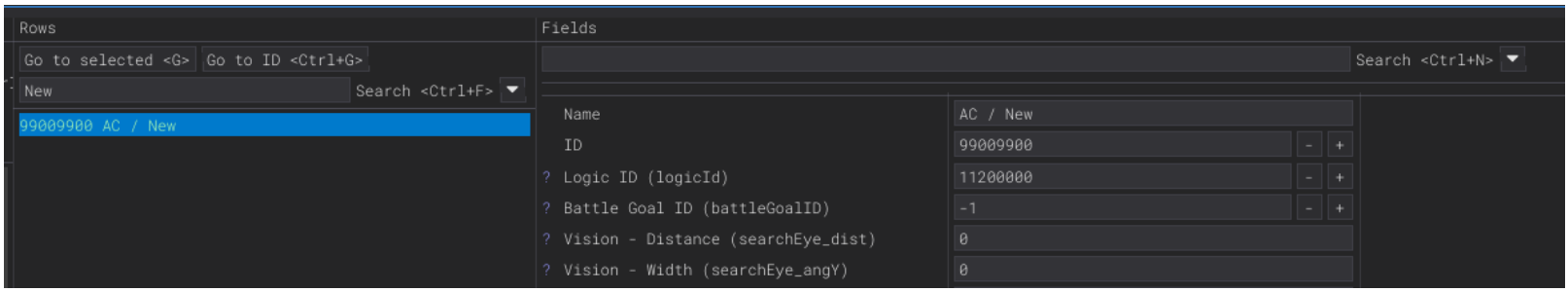
We're doing NpcParam next since this one is extremely simple, and we just need to create a new one and assign it the new AccountParam ID we made (so 9900). Of course, remember to change the AccountParamID in the ArenaParam itself as well.
I made it ID 99009900 just to keep with the scheme. (Why not 9900000? Because it was taken.)



We have now changed the name, both on the Arena and in-game.

2.3) Creating a new NpcThinkParam.

Very simple - we copy an existing one, and assign it a new ID. To be consistent, we'll use 99000000 again, same as the new NpcParam. This is where you would change the logicID to give it a different AI. I'll cover how to add custom ones later.

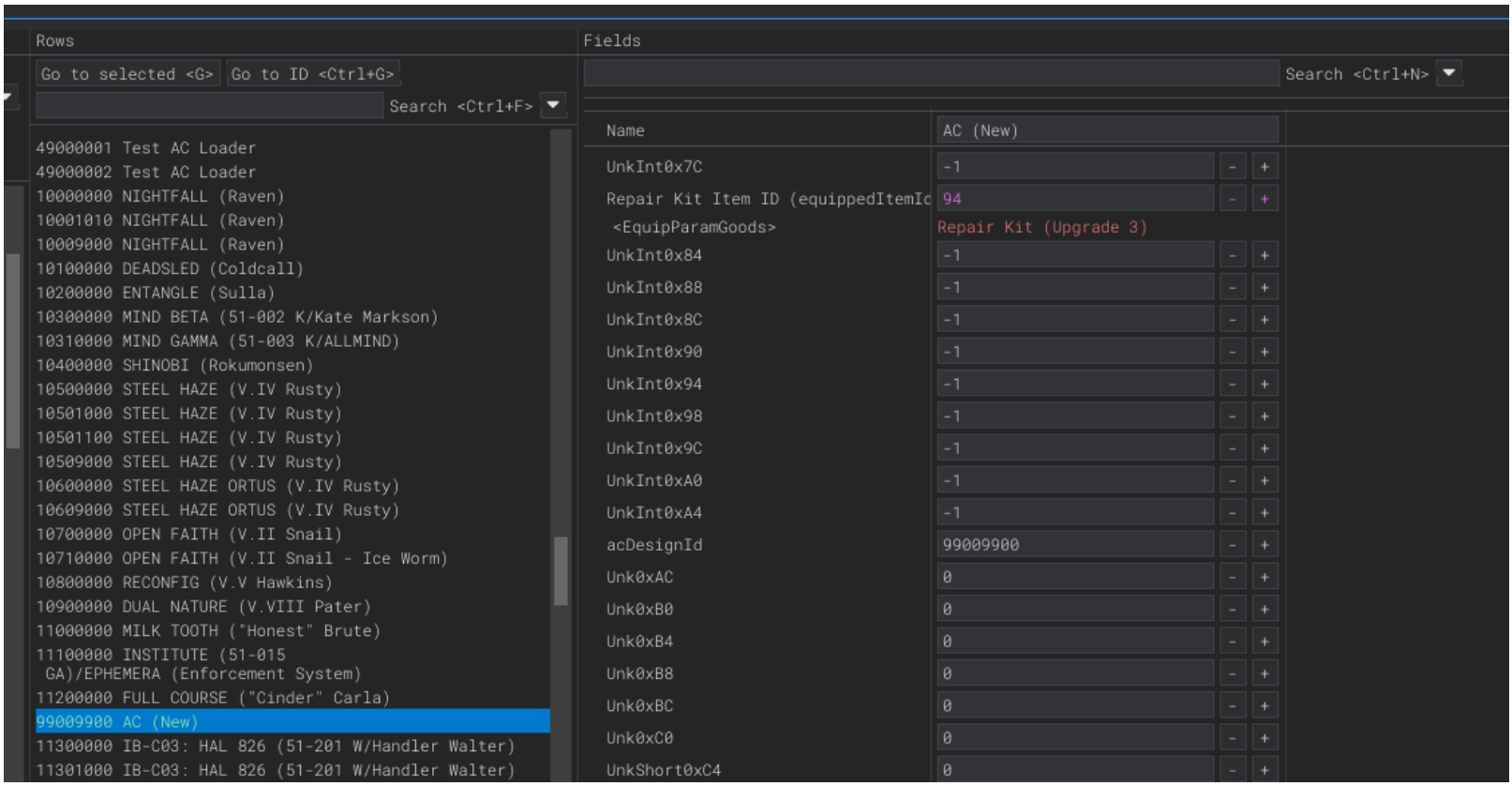


2.4) Creating a new CharaInitParam

This step is also very simple - we copy an existing one, give it the usual ID. This is where we start making more changes, though. This ID is what is used to reference the "Preview" image in the Arena, so it will disappear, since we haven't added one yet.

We will also set the only entry that actually matters, which is the acDesignId. This is what is used to reference a specific AC design file. I

We will also set the only entry that actually matters, which is the acDesignId. This is what is used to reference a specific AC design file. I will set this to 99009900, as usual, and we'll create and add the new file in the next step.



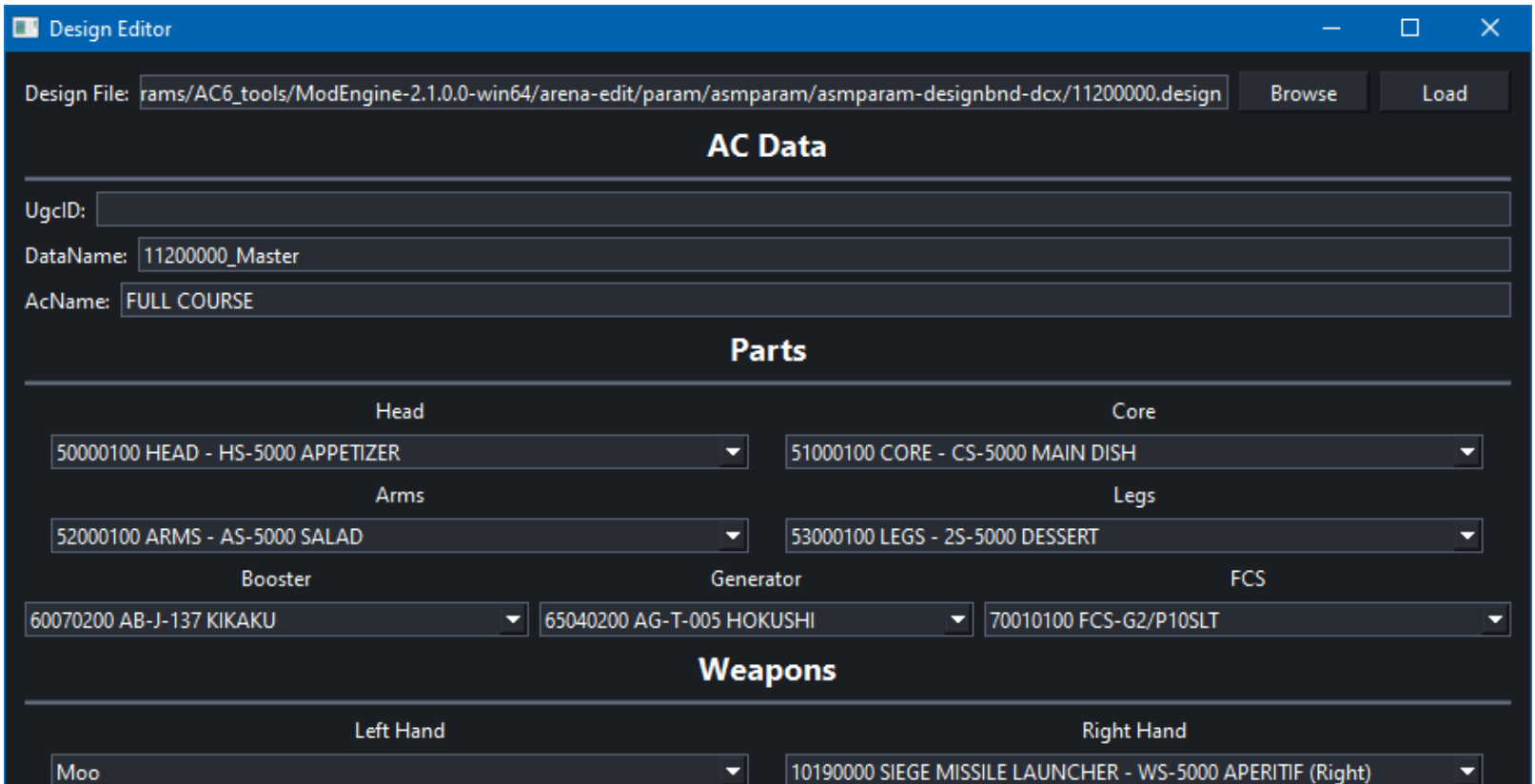
2.5) Adding a new .design file

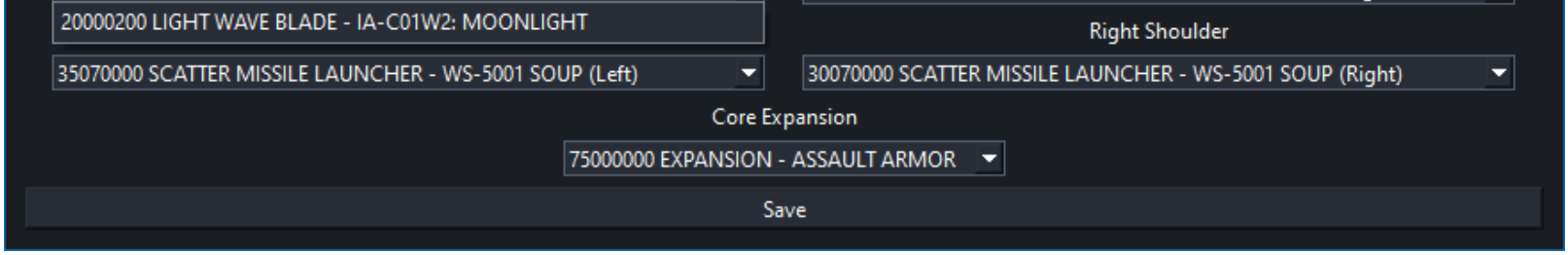
We will first create, in your mod folder (aka wherever you created the smithbox project) an "asmparam" subfoloder in param.

There, we will put (and unpack with WitchyBND) asmparam.designbnd.dcx, which is the bnd containing all the design files.

To create a new AC build, I would highly recommend using my .design editor. I'll open up a base AC (I'll pick Carla's, which is 11200000.design) and, I'll say, give her the moonlight. I will then save it as 99009900.design.

You *can* change DataName to match the new ID, but it's not necessary, and netiher is changing AcName, but it does look nicer.





Now that we've done that, we just save it as 99009900.design. Next, we need to edit the _witchy-bnd4.xml file present in this folder, so that when it's repacked, it will include the new design file.
Open it up in a text editor (I use vscode), go to the bottom, and add a new file entry with the ID being 1 higher of whatever the last ID in the list is, with the filename being the new design file.

This is an operation that will be done again later on to add the preview and emblem images, and I won't bother explaining it again.

Now we repack the asmparam.designbnd.dcx file. We now have an entirely new fight (since we can change both its AI and its AC design). We just don't have the preview image and emblems (unless we just re-use existing ones).

Step 3 - Image fixes

Okay. We've gotten all the functional bits working, but what good is a fight if you can't even make it pretty? So let's fix all the missing stuff.

Just like we did for asmparam, we need to copy a few files to the mod folder (and create the folder structure). Namely, menu/hi/00_solo_tpfbdt, menu/hi/01_common_ablythnd.dcx and menu/hi/01_common_tpf.dcx

hi/00_solo.tpfbdt, menu/hi/01_common.sblytbnd.dcx and menu/hi/01_common.tpf.dcx

3.1) Adding a new decal thumbnail (Included next to the HP bar)

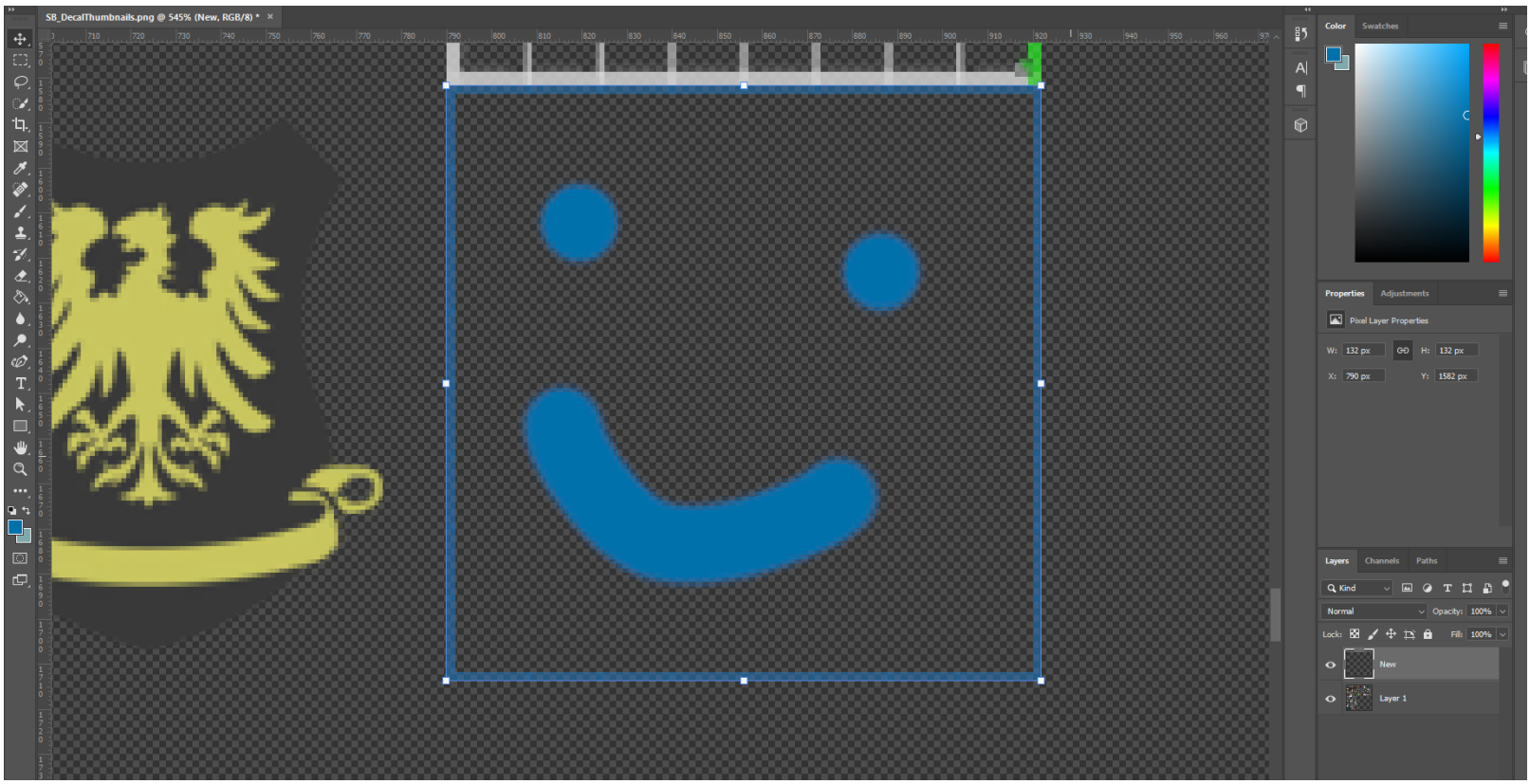
This one is kind of annoying. You need to unpack both 01_common.tpf and 01_common.sblytbnd.

In 01_common.tpf, you'll find the SB_DecalThumbnails.dds file. Personally, I use paint.net to convert it to PNG to open it in photoshop, and to then convert it back.

The file is split up into a grid of 132x132 squares. This includes a 2 pixel border on all sides, so in reality, the emblems are 128x128. What you want to do is add your new decal thumbnail in an empty spot. For example, you can copy the one that's a test pattern grid, and move it 132 pixels down, then erase everything but the 2 pixel wide border. And you'll have a good reference of where to place your pattern.

Once you've done this, take note of the X and Y pixel position of the top left corner of your new pattern. By that I mean the 128x128 square.

So for example, I know that the border of the new pattern (so the 132x132 square) is located at X = 790 px and Y = 1582 px.
I then add 2 to each value (to account for the 2 pixel border) and I get X = 792 and Y = 1584.



Now, just save it as png (then reconvert it to DDS) and repack 01_common-tpf-dcx. We're done with it.

Next, we need to unpack and look at 01_common.sblytbnd.dcx.
This file basically contains a bunch of .design files, which indicate the positions of subtextures within larger textures. In the case of SB_DecalThumbnails, we need to open up the SB_DecalThumbnails file in 01_common.sblytbnd.dcx with a text editor, and we'll see that it lists all the individual decals.

What we need to do is add a new entry to the TextureAtlas list. The name entry has to be name="Decal_tmb_0000[THE DESIGN ID].png", and the X and Y values are what we found earlier. Height and width are always 128.

So in our case, since we picked menuDecalId as 9900, we add this line:

```
147     <SubTexture name="Decal_tmb_00003044.png" x="792" width="128" y="924" height="128" half="0"/>
148     <SubTexture name="Decal_tmb_00003045.png" x="792" width="128" y="1056" height="128" half="0"/>
149     <SubTexture name="Decal_tmb_00003046.png" x="792" width="128" y="1188" height="128" half="0"/>
150     <SubTexture name="Decal_tmb_00009900.png" x="792" width="128" y="1584" height="128" half="0"/>
151     <SubTexture name="Decal_tmb_00009998.png" x="792" width="128" y="1320" height="128" half="0"/>
152     <SubTexture name="Decal_tmb_00009999.png" x="792" width="128" y="1452" height="128" half="0"/>
```

3.2) Adding a new Decal/Emblem, and a new Preview image.

This is much simpler. Unpack 00_solo.tpfbdt.
Additionally, move 00_solo.tpfbdh to the folder as well. We don't need to unpack and edit it manually, witchy will do so for us when we repack 00_solo.tpfbdt.

You'll see there are a BUNCH of files. What we're interested in are the files labeled:

- MENU_Archetype_10009000.tpf.dcx (These are the previews, the number is determined by CharaInitParamID)
- MENU_Decal_00000041.tpf.dcx (These are the emblems, the number is determined by DecalID)

So in our case, we'll want to create:

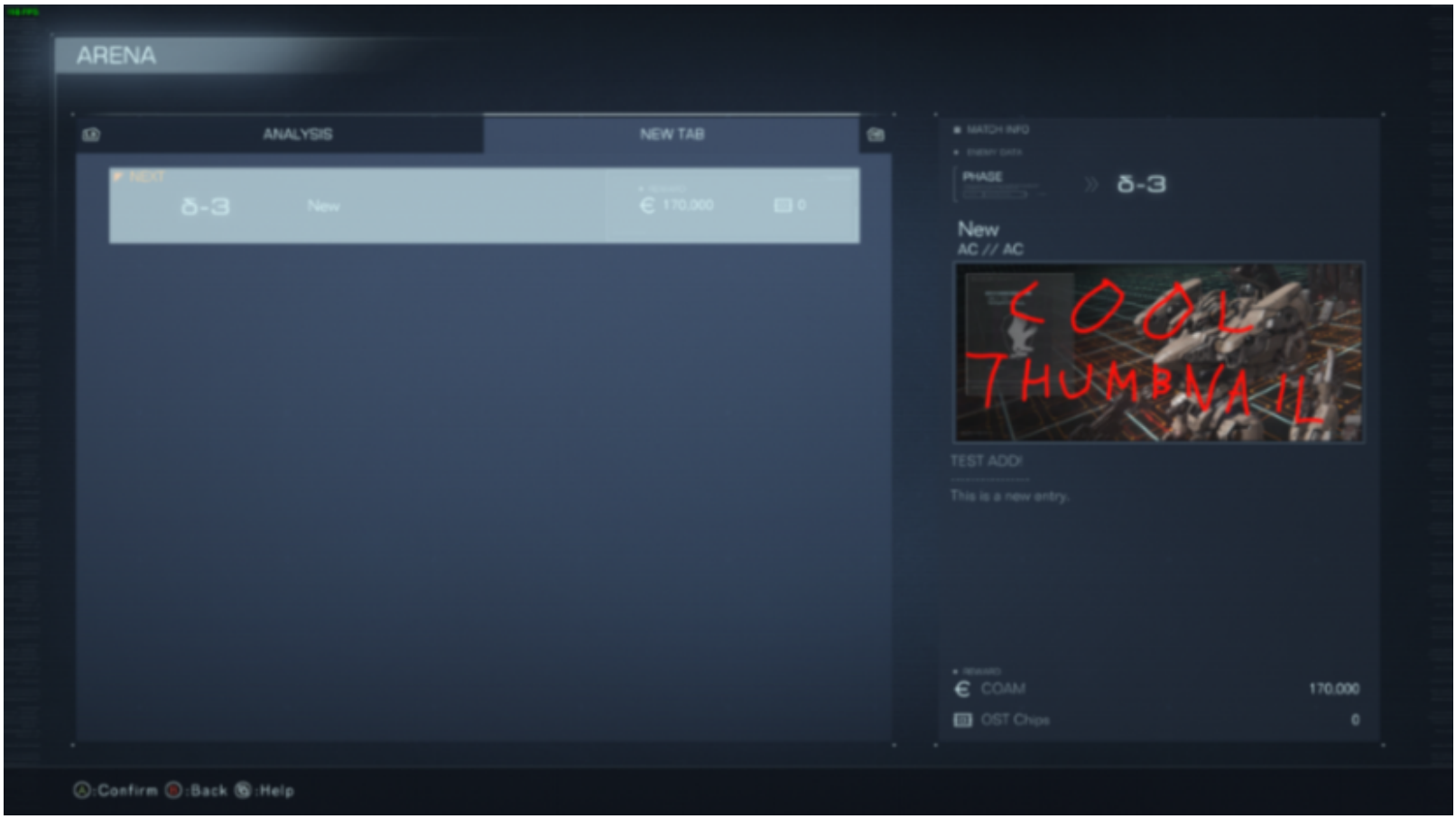
- MENU_Archetype_99009900.tpf.dcx
- MENU_Decal_00009900.tpf.dcx

We want to first unpack any existing Archetype/Decal file.
I suggest at this point renaming the folders to match the IDs we want - it's not strictly necessary, but it makes things more organized.

Next, we can edit the images inside them. First, we need to change their names so the IDs are the new ones, and then we can just make whatever edits we like.

Once that's done, we need to once again edit the _witchy-tpf.xml file for both of them. Namely, we need to change the IDs both in the resulting BND's filename, and the filename of the image itself.

Next you just repack them, and remember to add these new DCXs as new entries in 00_solo's _witchy-bxf4.xml file, just like we did before for the design file. And then repack 00_solo.tpfbdt as well.



Step 4 - Adding a custom lua think file (aka, a custom AI)

4.1) Copying an existing script file

Create a "script" subfolder in your mod folder. Then, find a think file, and copy it over.
It'll be called something like 11200000_logic.luabnd.dcx, which is the one for Carla. You can find the logicID in the NpcThinkParam.

4.2) Unpacking and editing the witchy XML

Unpack it with witchy. You'll see a file called 11200000_logic.lua - this is a *compiled* LUA script.
Grab a copy of DSLuaDecompiler (I like this fork by ElaDiDu, since it names some variables automatically) and run it on the script. It'll produce a file called 11200000_logic.dec.lua in the folder where the decompiler exe is located. Just copy it over to the dcx folder.

Rename it to match the new logicID you want to give this AI. To be consistent, I'll use 99009900, so now it's called 99009900_logic.lua - you can delete the original file. Make sure to rename the .luagnl file in the same way.
Next, edit the _witchy-bnd4.xml in the same way you did for the preview and decal images, changing the IDs in the files and the output file. But we're not done quite yet.

4.3) Editing the LUA file

The LUA file itself also contains the ThinkID, so we'll need to modify it.

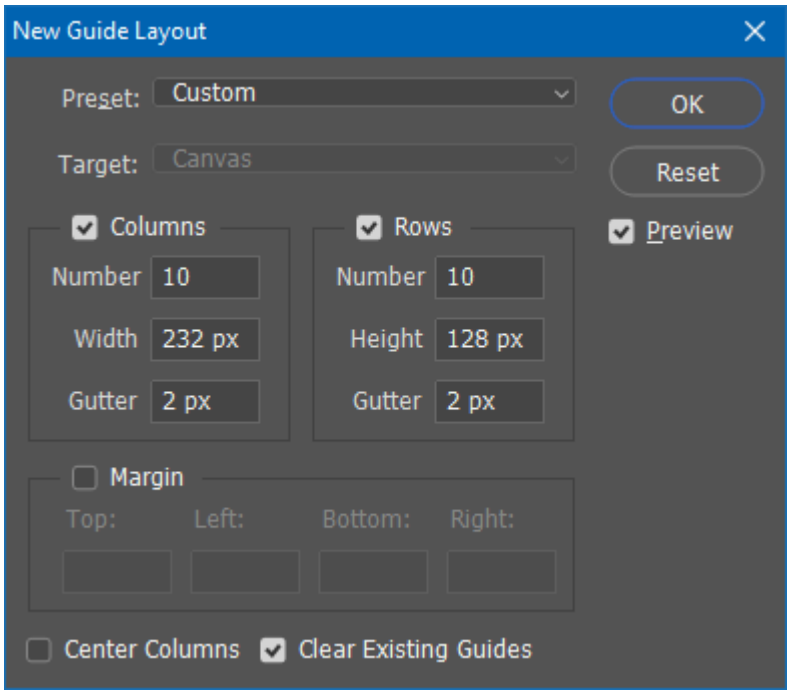
Just open it up in a text editor and replace all occurrances of the original ThinkID with the new one - in my case, 11200000 becomes 99009900. Save it, repack the DCX, and you're done. Remember to assign this new thinkID to the NpcThinkParam in Smithbox.

Step 5 - Adding Rank icons

Welcome to hell.

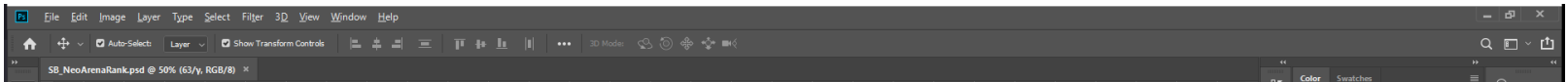
5.1) Creating the image containing all the rank icons.

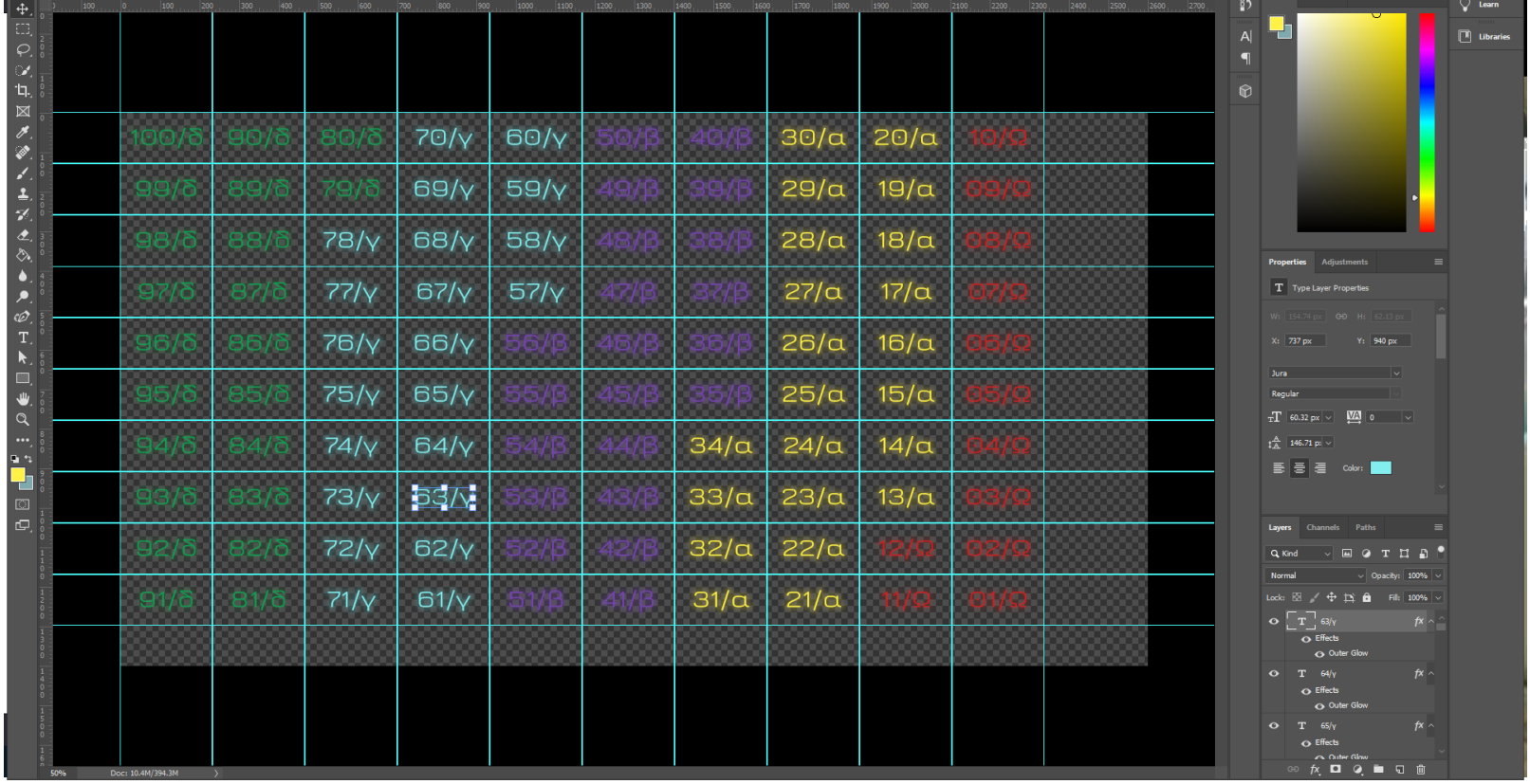
Pretty much what it says on the tin. Each rank icon is 232x128 with a 2px border.
I recommend using photoshop, as you can set up guides with the following settings (obviously the number of columns/rows should vary depending on the number of rank icons you want to add, this is for a 10x10 grid, so 100 images):



Take care to not leave empty cells at the bottom (as in, there must be _less than 128 pixels empty) - it's fine if there's more than 232px empty on the right side, though.

Here's an example image, with guides enabled:





5.2) Generating the .layout file

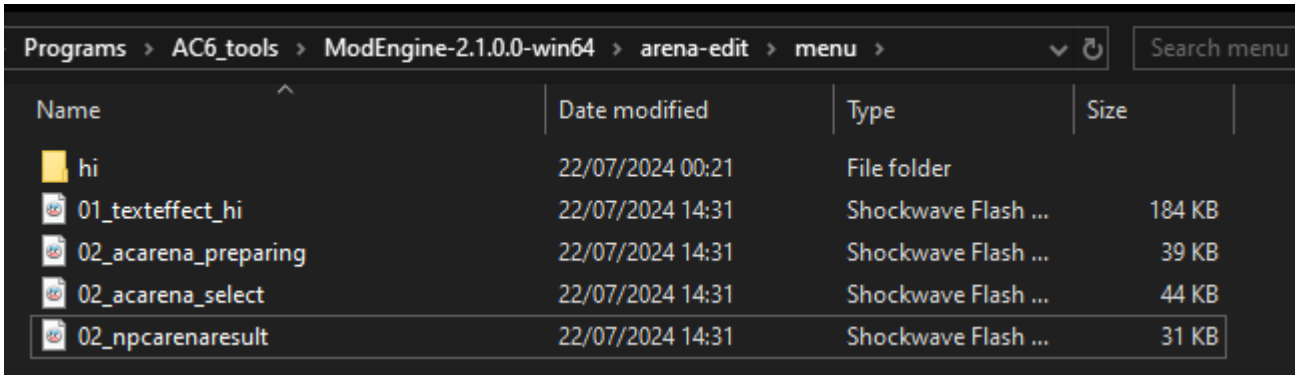
You'll want to download the generate_layout_file.py script from the github repo, and run it.
It'll ask for a few things, like a path to the image (I would recommend giving it the .png version, then converting it to .dds afterwards using paint.net - use BC7 Linear DX 11+).

It will generate a layout file for you. For both the rank image and the .layout file, you'll need to add them to the _witchy-tpf.xml file of the respective common file - that being 01_common.tpf.dcx and 01_common.sblytbnd.dcx

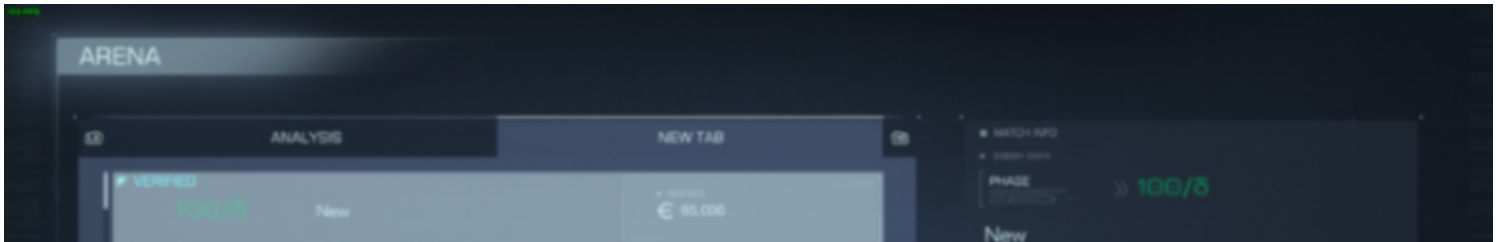
5.3) Adding your rank icons to the UI

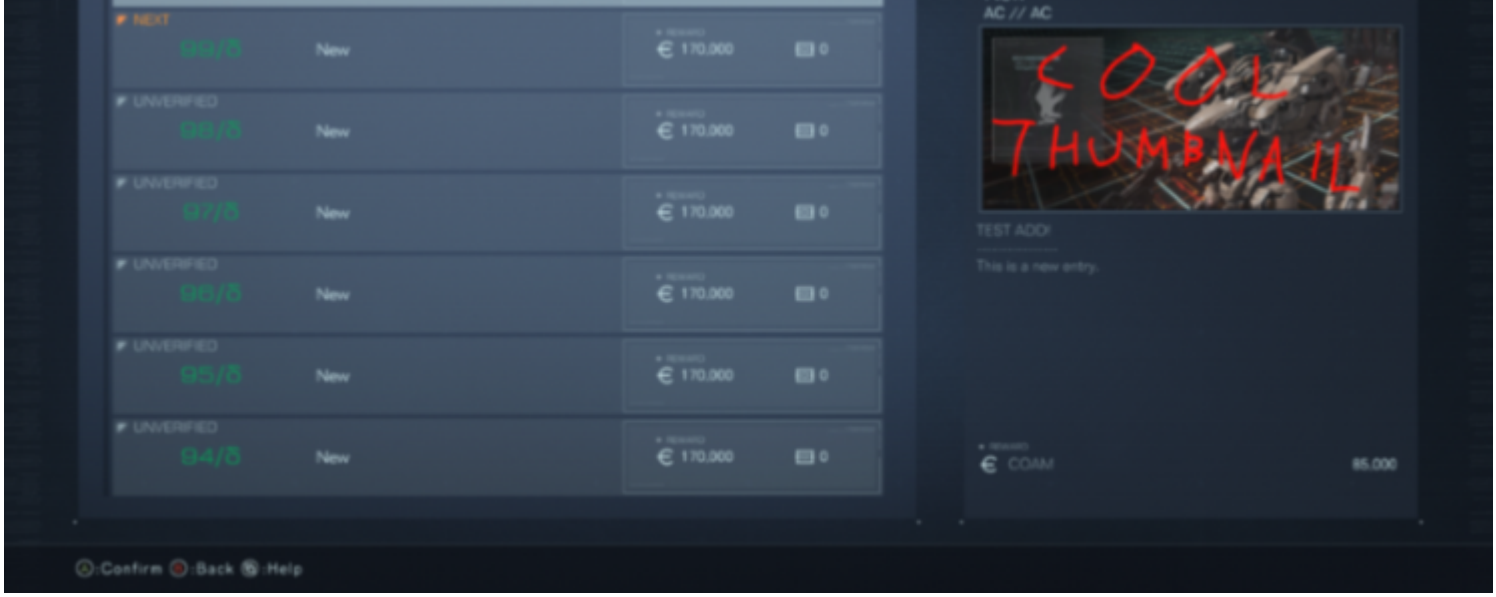
This is another step that will leverage a script I made, this time add_icons_to_gfx.py - it requires you to have JPEXS Flash Decompiler installed.

To prepare for it, you'll want to add these four gfx files in the "menu" folder:



Afterwards, you'll just need to run it, provide it the path to the layout file, then to all four of the GFX files, and it will edit the files for you.





Step 6 - Adding intro/outro dialog

Welcome to SUPER hell.

6.0) Calculating the intro/outro talk IDs

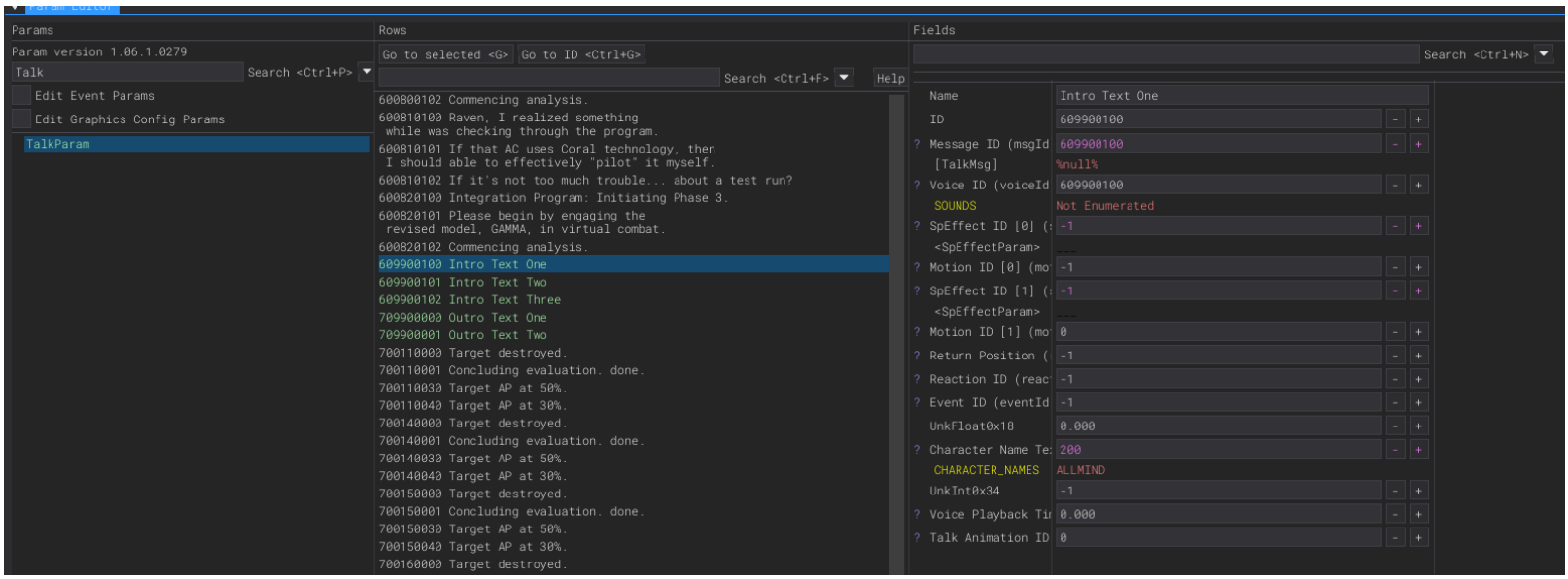
This step is very simple. The intro and outro IDs follow these formulas:

- Intro IDs: 600000000+AccountID*1000+[100/101/102, depending on whether it's the 1st, 2nd or 3rd line of dialog]
- Outro IDs: 700000000+AccountID*1000+[0/1, depending on whether it's the 1st or 2nd line of dialog]

6.1) Adding entries to TalkParam

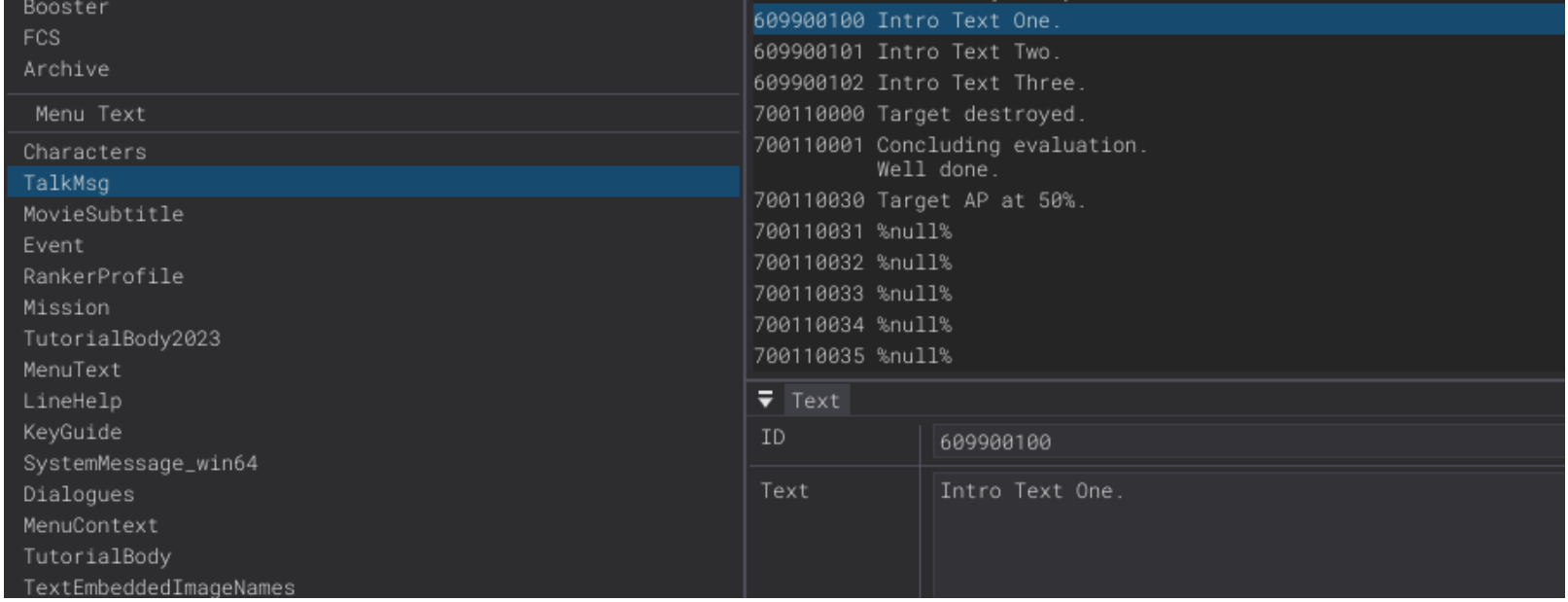
Just add an entry for each of the 5 IDs you've calculated. The ID, msgId and voiceId should all be the same one.

Pay attention to the Character Name - that will determine what character will be indicated as speaking the intro.



6.2) Adding entries to TalkMSG

These will be the subtitles displayed. They're referenced by the msgId in TalkParam, but the best approach is to just use the same ID.



That's it - you're done! You're done. There is no step after this. Stop reading. Please stop reading.

6.3) Adding custom audio

NOTE: YOU WILL NEED THIS BUILD OF MODENGINE2 TO REPLACE AUDIO FILES.

I will not be teaching you how to manually do it here. You should ask around in the ServerName discord's sound-and-dialogue channel.

I will instead be teaching you how to do it using a script I made, specifically add_voicelines_to_soundbank.py.
You will need to have wwise studio (2019+) installed, as well as having rewwise saved somewhere.

The script takes a file structure like:

- [AccountID]
  - Intro
    - 0.wav
    - 1.wav
    - 2.wav
  - Outro
    - 0.wav
    - 1.wav

And so on, for each AccountID you want to add.

You will need to have extracted the soundbank you want to add them to using bnk2json from rewwise. I recommend npc015.bnk as that is the one for ALLMIND, and is loaded at the correct time.
Do note that the script relies on another script, wem_conversion_wrapper.py, so I recommend just downloading the whole repo.

If you want to easily generate the audio file structure, you can look at generate_audio_tree.py.
This will generate it based on an exported TalkMsg. The example I include uses elevenlabs to generate it via TTS, but you could use anything else, like fetching the files from another folder.

Once you've run add_voicelines_to_soundbank.py successfully, you'll just have to repack the folder up as a bnk using bnk2json, move it to the correct spot, and you're done.