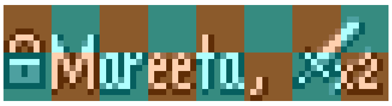
Menu text

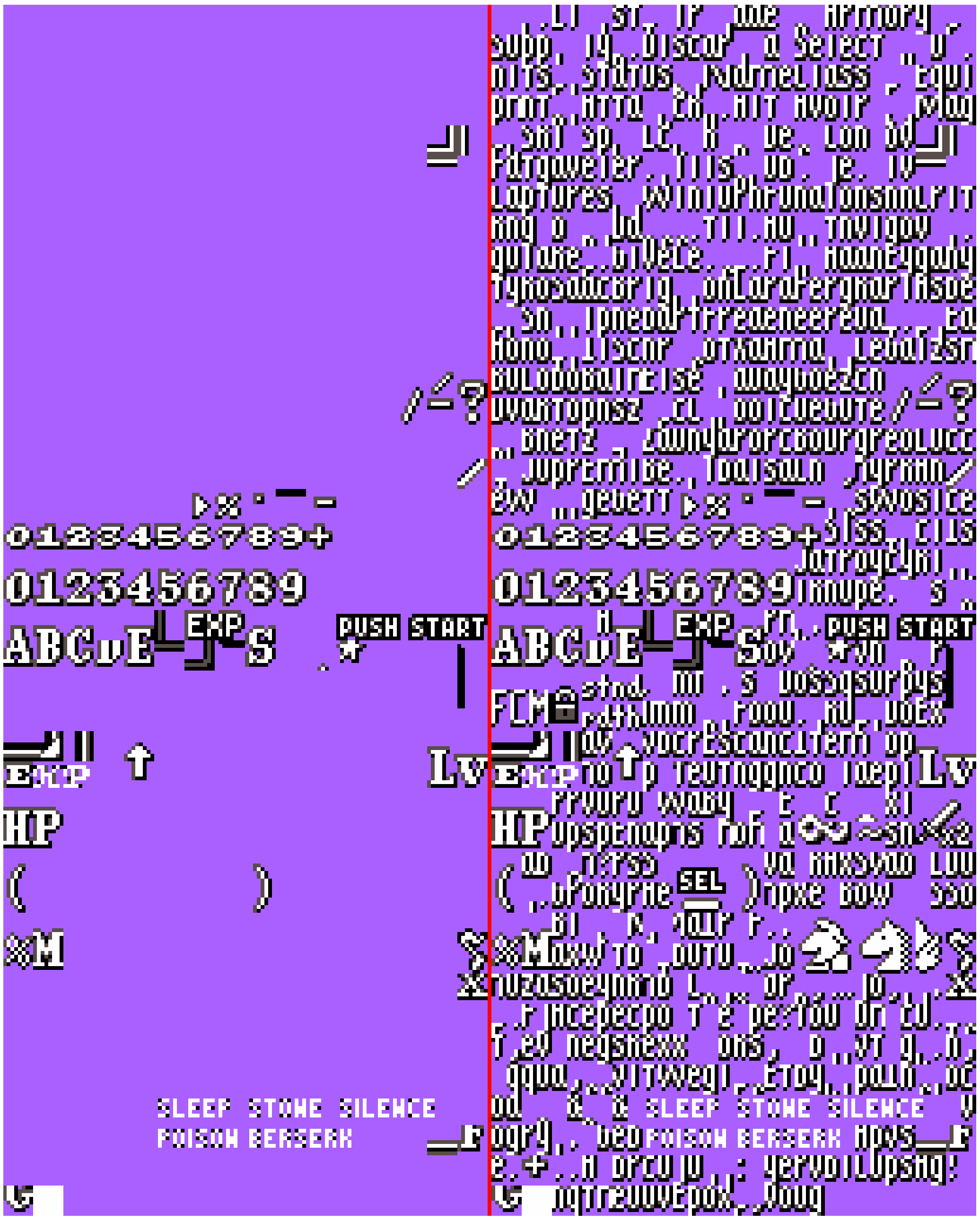
It might be surprising, but editing menu text (and editing it efficiently) is one of the more difficult aspects of modding/translating FE5.

Menu text is displayed in pairs of 8x8 pixel tiles stacked on top of each other. Like so:



Whether it's the top/bottom halves of a large letter, the halves of a pair of small letters (also known as a bigram), or a part of one of our icons, all of these tiles has to fit in a single block of graphics. This block also includes a variety of UI elements like numbers, operands, menu borders, etc.

To the left is our basic UI elements, to the right is what it looks like after all menu text has been included in it:



As you can see, space is pretty tight, and it's important to optimize your menu text if you want to fit everything in. Which is why our font is already optimized to make sure a large number of tiles can serve similar functions. For instance, the left side of the Cavalry effectiveness icon is the same as the left side of the Flier effectiveness icon. The tops of **S** and **C** are the same, and so are the bottoms of **a** and **d**, etc. But it all barely fits!

So, how do you get there in the first place?

1) Understanding the tools

Zane's menu text formatter, located in **text_menus\textformatter**, will generate this large block of text that the game will later use to make words that we can read. It does not care how you arrange these graphics, and simply generates them.

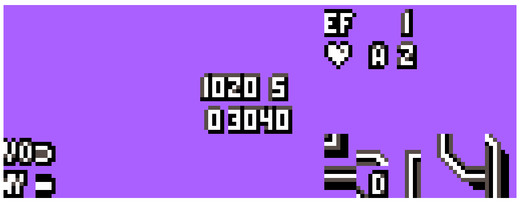
After that, the **text_menus** folder contains everything needed to arrange those tiles into words. It contains some tilemaps (**text_menus\tilemaps**), which some words will be manually baked into and a bunch of .asm files (**text_menus**), which the assembler will use to automatically arrange the tiles into words.

Think of the formatter as your dad going grocery shopping and making sure mom has all the ingredients for her meals. And the tilemaps/.asm files are the recipes.

2) Fixed tiles

A fixed tile is a tile that you will always use "as-is." It's the plus icons, the numbers, the parenthesis, the various effectiveness icons. If you need to add, edit or remove some of these tiles, simply edit **Vanilla.png**.

Keep in mind that the portion of the image below the 320 pixel mark can be edited (the terrain window is in there), but tiles added there cannot normally be used.



If you add or remove fixed tiles, you'll also need to edit the **FixedLayout.txt** file.

Case a: Tiles that you do not intend to call in a text string yourself, like menu borders or UI elements such as FCM, can be defined simply with

```
fixed     (X, Y)
```

where X and Y are the coordinates of the top left pixel of the tile. For instance, (0, 0) is the top-left corner tile, (120, 0) the top right corner tile and so on.

Case b: Tiles that you do intend to call in a text string yourself, like numbers, operands or special letters, can be defined with

```
string  "{Z}"  0xAAAA  [(X1, Y1), Xflip, Yflip, C]  [(X2, Y2), Xflip, Yflip, C]
```

Here, you can define a shortcut that can be used later to call forth a top-bottom pair of tiles, a word encoding for it, the coordinates of the top and bottom tiles, whether they're flipped horizontally or vertically, and what color they have. For more information on how to define these tiles, do consult the formatter's **ReadMe.txt**.

3) About generated tiles

When you build the ROM, your build file will call for the text formatter to generate new tiles for the menu text. This process involves three files:

- **Font.png**, which has the graphics for all the letters in the menu text font. This can be edited to change, add or remove characters, and can be vertically expanded at will.

- **Font.txt**, which tells the formatter which letter on your keyboard corresponds to which letter graphics on the png. So if, say, you change the **u** to a **µ** on there, congrats, you'll now need to type **Fergus** as **Fergµs**. You'll need to edit this file if you add more letters to the font.

- **TextEntries.txt**, which tells the formatter which letter pairs (bigrams) to make in the first place. This is daddy's grocery list.

If you type **Leif** in it, the text formatter will generate the top and bottom of **Le**, and the top and bottom of **if**.
If you add **Lara** to it, it will generate the bottom of **La** and **ra**. It will leave out the top of **La**, because it's the same as the top of **Le**. All that to say that the formatter checks for redundancies.

While most mods and translations will not require a lot of messing with fixed tiles or the font, basically every project will need to frequently edit the TextEntries file.

3) Generated tiles for text that's baked into a tilemap

/!\ 99% of English-speaking modders will want to skip this section. Translators, read on.

Some menu text is, unfortunately, not displayed through a regular **.text** string. and has instead been baked into a menu tilemap, occupying the same files as borders and other UI elements. This kind of text has to be manually edited into said tilemap.

Since the order of the words in the TextEntries determines the tiles' location in the output graphics, you'll want to keep all your baked words at the top of the list. Otherwise, changing non-baked text might dislocate tiles meant for baked text.
For similar reasons, you'll want to minimize edits to the baked text list, especially after you've already entered them into the tilemaps.

[**How to add that text to the tilemaps** is missing, because while it's fairly simple (if tedious) to do, it's kind of a pain to explain. Maybe I'll record a video of it or something. Until then, if you need your tilemaps edited for a translation, simply contact me on Discord and I'll do them for you.]

4) Generated tiles for normal text

Here lies the meat of the menu text work. In the TextEntries, all the words below the baked list will be used in **.asm** files, which will dictate to the game what words to use and where.

The assembling process goes

```
TextEntries    –formatter→  Tiles       –assembler→ }
                                                    } Words in the game
                            .asm files –assembler→  }
```

However, doing things in this order should be much easier for your work flow:

Fill up the ASM files → Update the TextEntries to match.

a) The menu text ASM files

These files are pretty easy to fill up. Everything is labeled, and most all character limits are indicated. All the menu text goes between quotation marks, after a **.text** tag.
Menu text that can support multiple lines (like item or skill descriptions) uses one **.word $0000** to indicate a new line, and two of them to indicate the end of the description.
Menu text that doesn't support multiple lines uses one **.word $0000** to terminate the string.

Like so:

```
_Vulnerary_name
.text "Vulnerary "
.word $0000

_VulneraryDesc
.text "Restore all {H}{P}"
.word $0000
.text "to the user "
.word $0000
.word $0000
```

However, there are a few additional rules to it. Remember that, as far as the game is concerned, there are are no individual characters, only tiles that can contain up to 2 characters. This will affect everything you write with menu text.

Rule one: You cannot end a line on a half-tile.
In the item name above, note the space after **Vulnerary**. That's because there are 9 letters, so your tiles are for the bigrams **Vu**, **ln**, **er**, **ar**, and **y**+**space**.

Likewise, Leif is 4 letters, and needs to be written as `.text "Leif"`, with no space.

Rule two: The shortcuts established in Step 2 take a full tile and cannot begin on a half tile.
In the item description above, **Restore all (+ space)** is exactly 12 characters, or 6 full tiles. Had this description been written as "Restores all HP", the text string would have required an additional space before the {H}{P} shortcuts.

[Or, the smarter option, which is to just shift the {H}{P} to the bottom line.]

Do also note that **Restore all {H}{P}** is going to be 16 characters wide. 12 small characters + 2 shortcuts worth double.

Rule three: Lil' Manster's font has some peculiarities and you should try to follow them.

Some of our letters are 2-characters wide to help with legibility. **M**, **N**, **W**, **m** and **w** are the most common ones, though you can find the full list in **text_menus\textformatter\Combos.txt**.

This means that, whenever you would type **Mareeta**, you'll need to write it as **(Mareeta**, as the **(** is used to form the first half of the M graphic, and **M** is used for the second half.

/**!**\ If you are translating Thracia into a language that does not require full capitalization (if you write **Spada di ferro** rather than **Iron Sword** like in English), You might want to give yourself fixed tiles for **M**, **N and W**, since you will never have to place them on a half-tile.

Rule four: For your own sanity in lowering the tile count, use fixed tiles when you can.

For numbers, operands and common abbreviations like HP, it is highly advised that you use the existing fixed tile graphics, even if it means having to rephrase a few description so that they don't land on half-tiles.

"Restore all HP" creates 2 tiles: The top and bottom of **HP**.
"Restores all HP " creates 4: The top and bottom of **space+H**, and the top/bottom of **P+space**.
{H}{P} is already in the graphics and costs literally nothing to use.

b) Updating the TextEntries

Once you've entered all your new menu text in the ASM files, it's time to update the TextEntries. Everything that was added, removed or edited in an ASM file needs to be changed in the TextEntries file. (If mom doesn't update dad's grocery list when she changes her recipes, some ingredients will go missing. Or dad will buy stuff that doesn't get used.)

Lines that start with // are comments and will not generate any tiles. Feel free to add your own comments to make navigation easier.

Most of the time, all you have to do is copy all the text between quotation marks line by line and paste in TextEntries. Do not worry about Rule One for TextEntries. You can just write **Vulnerary** without adding a space at the end.

Shortcut tags, however, need to be removed for the TextEntries. Like so:

ASM files	TextEntries
`.text "(Mareeta"`	`(Mareeta`

`.text "Restore all {H}{P}"`
`.word $0000`
`.text "to the user "`
	`Restore all`
	`to the user`

`.text "Restore {1}{0}{H}{P}"`
`.word $0000`
`.text "{+} user's (Mag"`
	`Restore`
	` user's (Mag`

`.text "He{im}'s Scroll "`	`//Heim's Scroll`
	`He's Scroll`
	or
	`//Heim's Scroll`
	`He`
	`'s Scroll`

The comments in the last example aren't necessary (as most comments tend to be), but help with navigation.

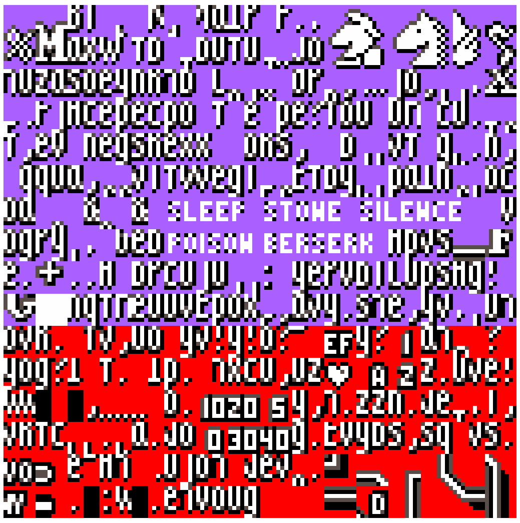
c) <u>Building</u>

If you've read the ReadMe in the root folder, you know how to build the ROM. Building will also run the formatter and everything, so it's always a good idea to build after menu changes, if only to quickly identify any potential mistakes.

Check the fe5trans ROM. Did it build? If not, check the log file. Chances are you made a mistake while filling out the ASM files, or when copying over to the TextEntries.

Keep troubleshooting until, at last, the ROM builds. Now check Osian's inventory. His Vouge is set at the end of the TextEntries, since it has unique bigrams. If you ran out of tile space, the Vouge's bigrams will be the first to go. Does the Vouge read properly? If it does, carry on to the next section.

If it doesn't, time to check how much tile space you went over.
Open **text_menus\textformatter\output\MenuTextFont.png** to witness your handiwork.
Everything in the red is going over.



(Note: You can also check the **MenuTextFont.png** file directly without opening the game and checking Osian's Prf. I just figured it was worth noting.

d) Help, I badly ran out of tile space

Yeah, it happens. All the time. Time to *optimize* things.
Aside from Rule Four mentioned above, there are no hard-and-fast rules on how to optimize your bigrams, just general advice, presented here with good/bad examples.

Keep your phrasing consistent

No	Yes
_KillingEdgeDesc .text "High critical rate"	_KillingEdgeDesc .text "High critical rate"
_VougeDesc .text "{Lock}Osian " .word $0000 .text "Can attack at range " .word $0000 .text "High critical rate"	_VougeDesc .text "{Lock}Osian " .word $0000 .text "Can attack at range " .word $0000 .text "Increases crit rate "

Re-use chunks

No	Yes
_VeninEdgeDesc .text "Poisons the target" .word $0000 .text "on hit"	_VeninEdgeDesc .text "Poisons the target" .word $0000 .text "on hit"
_JormungandDesc .text "Potent dark [magic" .word $0000 .text "Ene[my [mages]will" .word $0000 .text "also poison the" .word $0000 .text "target on hit"	_JormungandDesc .text "Potent dark [magic" .word $0000 .text "Poisons the target" .word $0000 .text "on hit if used" .word $0000 .text "by ene[my [mages"

Avoid landing capital letters and punctuation on half-tiles

No	Yes
_BeoBladeDesc .text "{Lock}Fergus, Diar[muid "	_BeoBladeDesc .text "{Lock}Diar[muid, Fergus"

Be smart with your line breaks

No	Yes
_TorchDesc .text "Gives {+}{1}{0}" .word $0000 .text "vision in fog" .word $0000 .text "of]war. Effect" .word $0000 .text "decreases by" .word $0000 .text "{1} every turn "	_TorchDesc .text "Gives {+}{1}{0}" .word $0000 .text "vision in fog of" .word $0000 .text "]war. Effect" .word $0000 .text "decreases by {1}" .word $0000 .text "every turn "

And if this isn't enough to fit everything in that poor little ROM, you can also make some sacrifices:

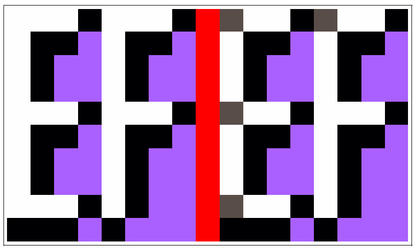
- Minor aesthetic touches like the “Middle-size end-of-word w” (written as **12**) or the **{im}** and **{mm}n** shortcuts can be replaced with a more standard **]w**, **i[m** and **[m[m**, although you have to be wary of character count changes, for the latter two.

- Unused characters, classes and items + descriptions that you might not be using in your project.

- The Sound Room, since it's an obscure and difficult menu to reach, with 12 unique bigrams. You can replace all category and track names with “Category {1-9}” and “Song {1-9}{1-9}” if you want to be fancy. Or just remove them altogether and never look back.

- Options descriptions. Not the worst thing in the world to have missing.

- Change your font file to have your E and F as such:



This simple optimization is guaranteed to save a frankly outrageous amount of tile space. However, it will negatively impact the looks of the font and require you to redo the tilemaps with menu text baked into them. Consider this only as a last resort.

e) <u>Congratulations</u>

If you've gotten this far, and have changed more than just the bare minimum of menu text (looking at you, translators), then congratulations, you've surmounted the most difficult part of editing FE5. Know that everything after this is a smooth ride... well, kinda. Depends how deep you wanna go.

If you have any further questions on the topic, don't hesitate to contact me (Miacis) on Discord.