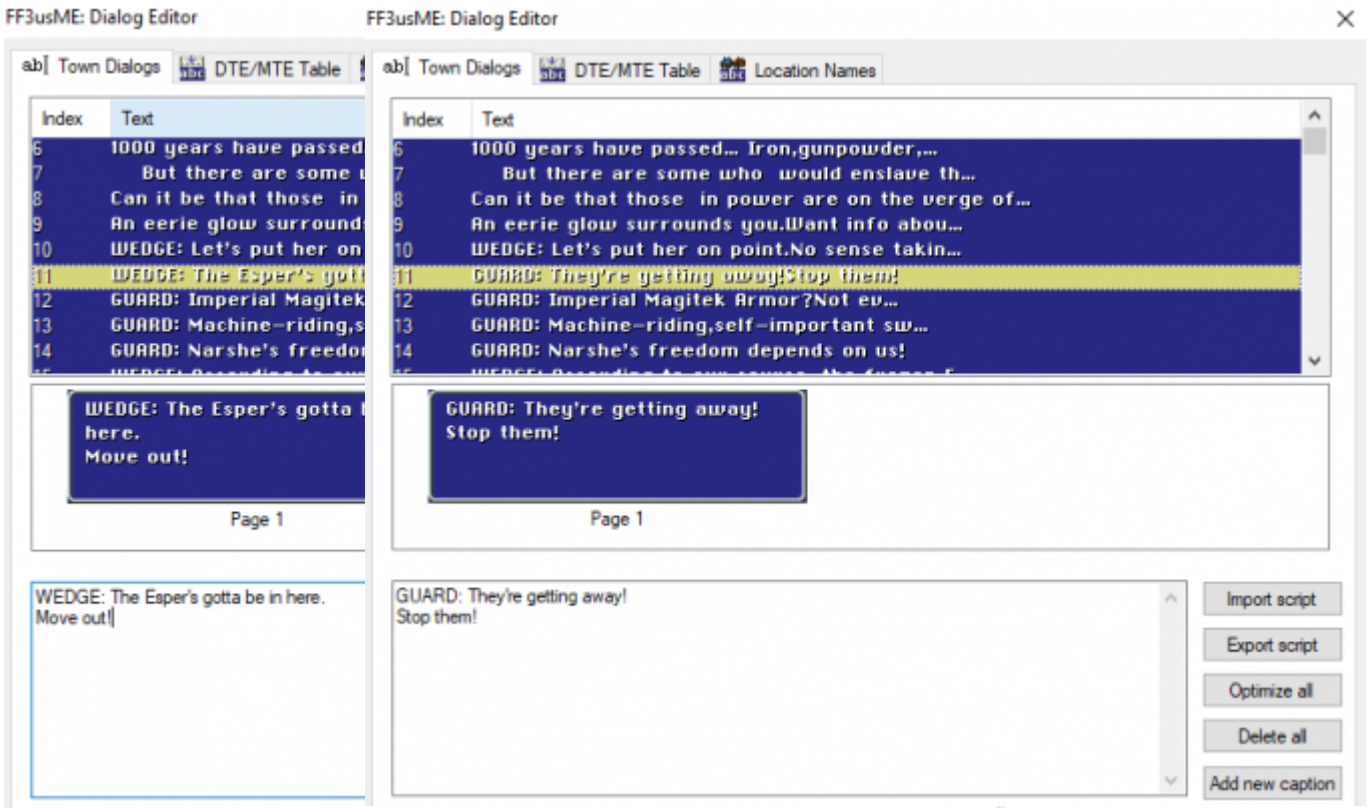
First Event

Alright, let's actually dive into this now. To practice eventing, we probably want to modify an event that occurs near the beginning of the game (for the sake of convenience). Let's try modifying the event that occurs if you try to leave Narshe at the beginning of the game. Normally, Wedge would yell “The Esper's gotta be in here. Move out!”, and you'd turn around to walk back into Narshe. Let's try modifying this event so that Narshe Guards attack you if you try to leave. To find the event in the event script, follow the process from before: select the event tile in the LE, note that the event is at 29B1D, and convert it to CC/9B1D. Then find the event in the ROM by searching for C9B1D.

Display a Text Box

Now we can start modifying this event. We should probably notify the player of what's happening, which can be be accomplished by showing them a text box. We can change text boxes in FF3usME, as you may have seen already. But which text box should we modify? Since we're removing the original cutscene that played here, we know that “The Esper's gotta be in here. Move out!” can be changed to our new text. Open the dialogue editor in FF3usME and modify the text to something fitting. Press Apply and OK to save the text, then press Save in the main window to write the new text to the ROM. Now we have a text box to display.



The next step is to figure out which parameters will display this particular text box. In FF3usME, you can see that the text box we modified is at index 11, which is highlighted in the images above. Remember that 11 is in decimal, but our parameters are in hexadecimal. Looking at the Event Command Document, we can see that we will need to convert *12* to hexadecimal if we want to display the text box at index 11. In hexadecimal, this is $C, or $000C. Now we flip this around as we did previously to get 0C 00 as our parameters. I see no reason to either move the text box to the bottom of the screen or remove the background, but you can play around with that if you want. Therefore, the full command to display this text box is 4B 0C 00. We can type this into HxD at the address we found previously, which gives us the following result:

```
000C9B00  FE F4 D1 55 80 94 D2 B5 D2 33 D2 BF 4B 0A 00 B6  þôÑU€”ÒµÒ3Ò¿K..¶
000C9B10  18 9B 02 B3 5E 00 3A FE 4B D4 06 3A FE 4B 0C 00  .›.³^.:þKÔ.:þK..
000C9B20  0F 00 82 D7 FF 3C 00 FF FF FF 41 00 45 3D 0E 3D  ..‚×ÿ<.ÿÿÿA.E=.=
```

Perhaps you want to take a look at the event as it is now. If nothing else, you might want to check that the event functions properly. Every event ends with the FE command, which tells the game to return from the current event. So finish the current event with the FE command, and save it in HxD. Now you can open it up in your favorite emulator and see the text box in action.

```
000C9B00  FE F4 D1 55 80 94 D2 B5 D2 33 D2 BF 4B 0A 00 B6  þôÑU€”ÒµÒ3Ò¿K..¶
000C9B10  18 9B 02 B3 5E 00 3A FE 4B D4 06 3A FE 4B 0C 00  .›.³^.:þKÔ.:þK..
000C9B20  FE 00 82 D7 FF 3C 00 FF FF FF 41 00 45 3D 0E 3D  þ.‚×ÿ<.ÿÿÿA.E=.=
```

You may notice that the text box is displayed over and over again. This is because Terra is still standing on the event tile when the event ends, so the event is triggered again after it is done. We don't have the tools to fix that yet, so we will ignore the issue for now.

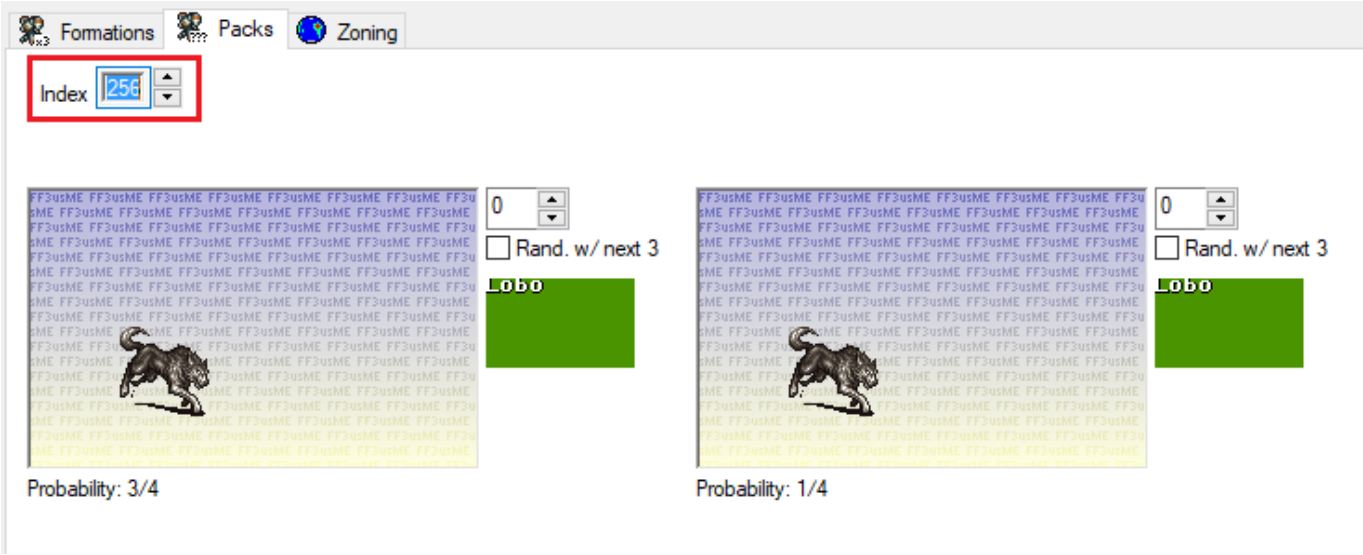
Red Flash

How about we add in a red flash, to show that something dangerous is coming? If we search for the word “flash” in the Event Command Document, we quickly come across command $55. The document tells us that there is only one parameter, which controls the color of the screen. To get a red color, the parameter should be 20. So our full command to flash the screen is 55 20.

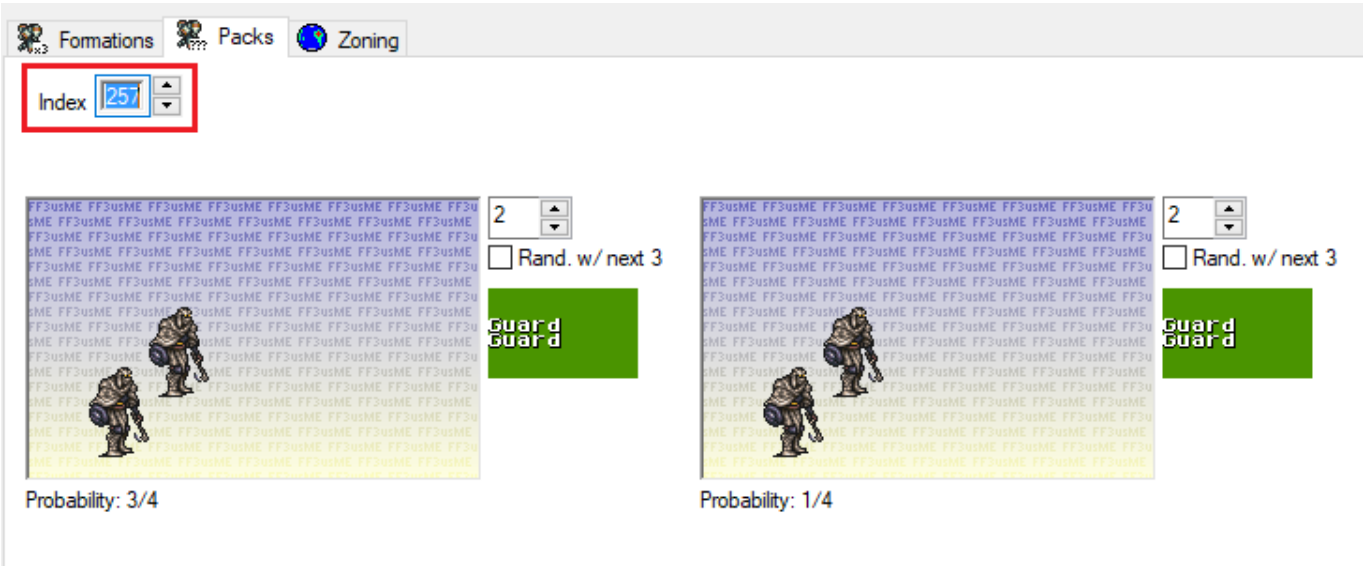
Force a Battle

Finally, we want to end the event by forcing the player to battle some guards. Searching for the word “battle” in the Event Commands Document, you may come across command $4C. This command is obviously quite involved, and there are some extra notes explaining information that might be useful. In particular, NOTE1 tells us that we should be using command $4D to call a battle here, although we still need to look at this section for information about the parameters. Ignore NOTE2-NOTE5 for now; we don't have enough knowledge to decipher them yet. Read over the other two notes for yourself.

This command has two parameters, like command $4B. Looking at EX1 and EX2, it looks like the first parameter controls which formation is encountered. The Event Command Document is directing us to look in FF3usME again. This time, we want to look under the “Packs” tab of the Formation Editor. If the first parameter is $00, it corresponds to formation 256 in this editor. This means that the player would encounter a single Lobo if we set the parameter to $00.



From the second example, we can see how to specify the encounter that we really want. The parameter is being added to 256, and the result is the formation that will be fought by the party. If the parameter is $01, the party will fight formation 257. If it is $02, the party will fight 258. If it is $19, the party will fight 256 + $19 (which would be 256 + 25 = 281 in decimal). Now we just need to figure out which formation contains two guards. Thankfully, we don't need to look very far, as this is formation 257. Therefore, the first parameter should be $01.



The second parameter controls the background of the encounter. EX1 shows us that each background corresponds to a specific ID that we could look up in FF3usME. However, EX2 gives us a better alternative in this case. The parameter $3F is special, as it instructs the game to use the background for the area that the party is currently in. Since we're in Narshe, it will use the Narshe background. That is exactly what we want, so let's set the second parameter to $3F.

According to NOTE7, we could add another value to disable the usual mosaic effects and sounds that occur when a battle is entered. I see no reason to change this, but you can play around with it if you want to.

Putting all of this together, the full command that we need to use is 4D 01 3F. Now our entire event looks like this:

```
000C9B10  18 9B 02 B3 5E 00 3A FE 4B D4 06 3A FE 4B 0C 00  .›.³^.:þKÔ.:þK..
000C9B20  55 20 4D 01 3F FE 00 FF FF FF 41 00 45 3D 0E 3D  U M.?þ.ÿÿÿA.E=.=
000C9B30  0F 0E 86 D5 26 32 CE C2 FF 0F 86 D5 26 32 CE C2  ..†Õ&2ÎÂÿ.†Õ&2ÎÂ
```

Now it's time to look at the event in the emulator. At the moment, the event does what we told it to, but it's still rough around the edges. The most obvious issue is that the screen remains black at the end of the event.

Troubleshooting

Restoring Screen from Fade

It's unsurprising that the event is not working perfectly on the first try. We need to work out what went wrong and how to fix it. You may have already worked out the solution from the information you've seen up to this point, but it might be good to look at some troubleshooting tactics anyways.

There are a couple different places that we could look for information. One tactic we could use is looking at the Event Script Dump and finding other examples involving the $4D command. Then we might notice something that we're missing. If you search for “ 4D ” in the Event Script Dump (include the spaces in your search), you'll find plenty of different battles that are fought during events. If you're observant and/or lucky, you might notice that the $96 command is used after many of these

battles. This command “Restores screen from fade”, which is what we want to do!

Admittedly, searching through the Event Script Dump wasn't particularly effective in this case. First, we don't know what the $B2 command does yet, so it required extra guesswork to determine that we wanted to look at the $96 command *below* that. Second, it turns out that the necessary information is also available in the Event Commands Document. You should still remember that looking at the original game's script can be a great way to solve problems or learn new things.

Alternatively, this information is available in the Event Commands Document. Looking back at the $4C command, we can see that we forgot about NOTE6! In order to fade in the screen here, we need to use either the $59 command or the $96 command. The only difference between them is that the $59 command allows us to specify what speed we fade in the screen at, while the $96 command has no parameters and always fades in the screen at the same speed. You could use either command, but as we noted earlier when looking at the Event Script Dump, the game usually uses the $96 command to fade in after battles. We shall do the same.

Delays

The next issue is more of a stylistic one. Usually, a flashing effect doesn't occur at the same time as a mosaic effect. I think that there are too many visual effects on the screen at the moment when the battle is about to begin. You may disagree (and you are certainly free to do whatever you want), but I want to prevent the flash from overlapping with the mosaic effect. To that end, we can put a delay in the script. Each delay pauses the execution of the event script for a certain number of “units”. These units of time are very short. Rather than try to calculate exactly how long you should pause the game for, I recommend simply playing with the delays until you get a result that *feels* right.

The simplest delay commands are $91 through $95. One of these will generally be sufficient for most purposes. You can experiment with them all if you want. The final event will look something like this, where the highlighted value is replaced by the delay that you are using:

```
000C9B10  18 9B 02 B3 5E 00 3A FE 4B D4 06 3A FE 4B 0C 00  .›.³^.:þKÔ.:þK..
000C9B20  55 20 9F 4D 01 3F 96 FE FF FF 41 00 45 3D 0E 3D  U ŸM.?–þÿÿA.E=.=
000C9B30  0F 0E 86 D5 26 32 CE C2 FF 0F 86 D5 26 32 CE C2  ..†Õ&2ÎÂÿ.†Õ&2ÎÂ
```

Now this event is looking a bit more fleshed out! However, the event still repeats itself. To deal with this, we'll need to learn a bit about action queues.