Compressed Graphics Tutorial

In this tutorial, we will edit a compressed graphic with the help of four utilities. We will first decompress from the ROM the graphic, modify its palette in the ROM, edit the graphic and finally compress the new graphic in the ROM.

The Tools

The first utility you will need is Peer Sprite Viewer (PSV). This utility can compress and decompress data from the FF3us and Chrono Trigger ROM. Secondly you will need YY-CHR. There is a .NET version and a C++ version. I personally go with the C++ one; it is older, but it has more features that have not yet been ported to the .NET version. The image shown here is from the C++ version. The other utility required is SNESpal, which can modify color palettes in the ROM.

Finally you will need an all purpose image editor. My favorite one is Gimp for many things, including icon making, spriting and other graphics editing. Visit gimp.org for the latest version.
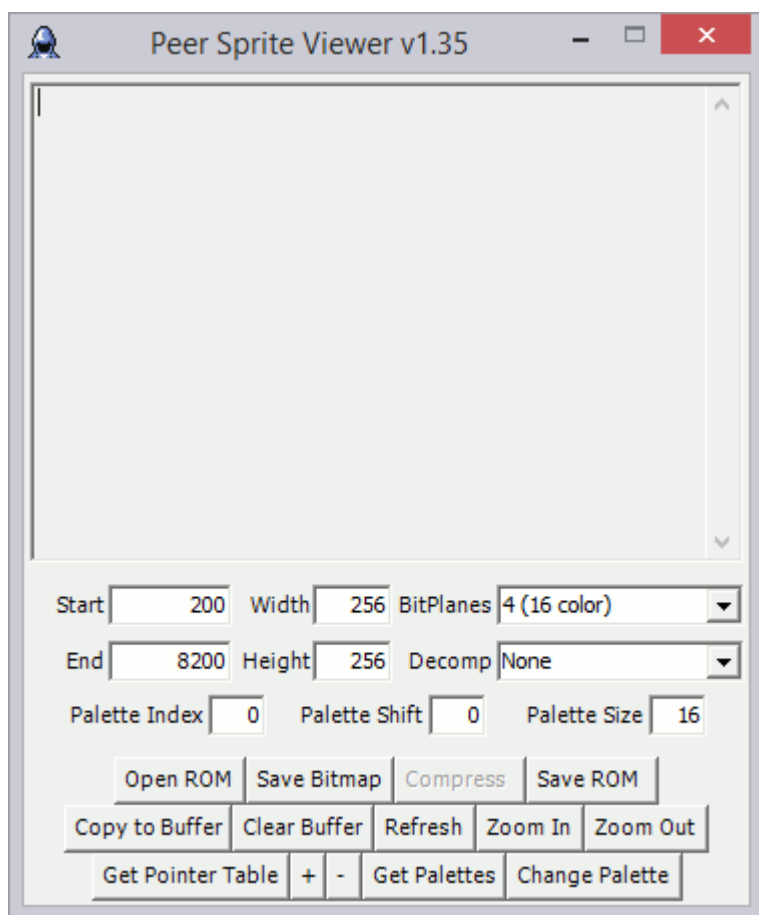
Decompressing the Graphic

The first thing to do is to locate what you need to decompress. We have a Compressed Data List which include graphics. Alternatively, you can look at the ROM Map but theoretically everything should be in the Compressed Data List. We will take for this tutorial the battle shield used in shield blocks by party members. It is part of a bigger *graphic sheet*, meaning you cannot just decompress / compress the shield alone. Modifying the shield will have an impact on the size and arrangement of the data once re-compressed.

The battle graphics we are looking for are at $D2E000 (unheadered ROM). This is $12E000 in absolute addressing. If you open a hex editor and go at this address, you'll see the following:

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0012E000   44 0B DE DD 1F 0C 04 0C 0C DD 0F 01 01 F0 DE 27
```

The first two bytes are always an inverted word of the compressed data size. This mean our graphics are $0B44 bytes in size (including the first two bytes), meaning they are from $D2E000 to $D2EB43. The wiki list the graphic ending at $D2EBFF, which is not totally false since $D2EB44-$D2EBFF is unused space. However, while the starting offset will always likely be good, I suggest you check the data size to decompress the exact amount of data.

Open your ROM in Peer Sprite Viewer (PSV). You'll see the main interface:

Peer Sprite Viewer v1.35

Start 200   Width 256 BitPlanes 4 (16 color)
End 8200   Height 256 Decomp None
Palette Index 0   Palette Shift 0   Palette Size 16

Open ROM | Save Bitmap | Compress | Save ROM
Copy to Buffer | Clear Buffer | Refresh | Zoom In | Zoom Out
Get Pointer Table | + | - | Get Palettes | Change Palette

Start is your starting offset, **End** is your ending offset, **Width** and **Height** are the wanted dimensions of your decompressed graphic, **BitPlane** is the type of graphic once decompressed and **Decomp** is the compression type. I have always kept the **BitPlanes** value as *4 (16 colors)* for FF3us and obviously **Decomp** value should be set as *Final Fantasy VI*. This screenshot show the correct values to enter for our example. Now click **Copy to Buffer**.

You should see the graphics in another window. Note that you can modify the sheet size and re-click on **Copy to Buffer** to get the right size. Here are the same graphics, first screenshot is default 256×256 and second one is 128×128. This does not change much, only it will be easier to edit if you got the correct size.
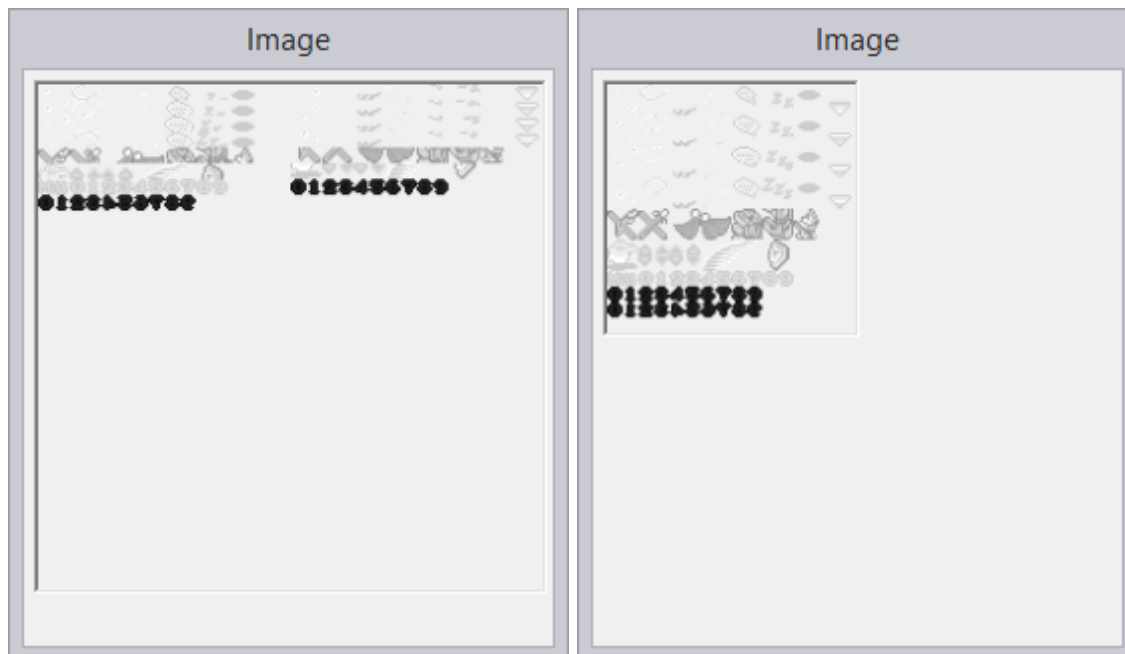
The next step is saving this graphic into a file. Click **Compress** and you will see this window:
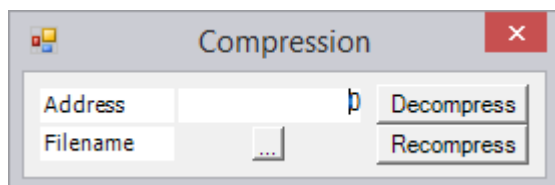

Enter the same starting offset ($12E000) and click **...** to enter a filename with the extension of your choice (a lot of people use .bin to hint at a binary file). Finally click **Decompress**, as shown here. Your GFX is now saved in a file we will next edit with YY-CHR.

Finding and Editing the Palette

Now we need to use **YY-CHR**. If you want the compressed files and correct savestate to load the palette, you can download them here. When you lad the decompressed GFX, it should look like this. No need to worry here, we will load the palette from a ZSNES savestate which was taken when the shield was on screen. Do *Palette → Load Emulator state file* and select your savestate. Click **Pal** at the bottom right window and try each palette until you get the correct one. It should now look like this:
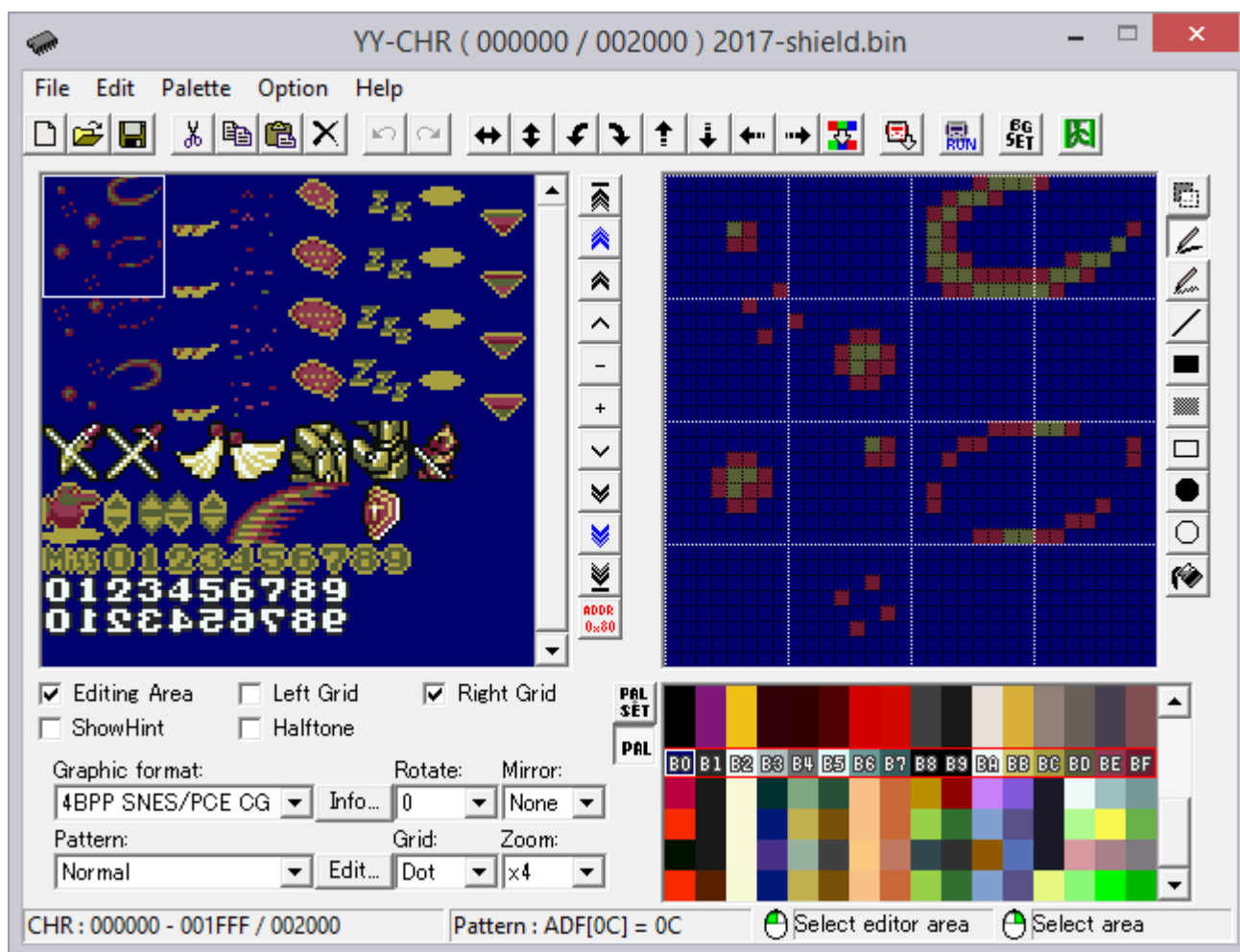
When you right click on a palette color you will see its RGB value as shown here. In this example, color BB RGB values are D0,C8,68 which is in hexadecimal notation. You'll need to convert the hex values into decimal (0-255) to find the palete with SNESpal. When entering the values in SNESpal, make sure you check the autoconvert checkbox. The example in the next picture is no with the last color of the palette. There is no real trick to find the color except looking at the search results and finding the few colors that are in same order as in YY-CHR. Try searching with "rare" or uncommon colors and do not search for black or white as it will give tons of results.
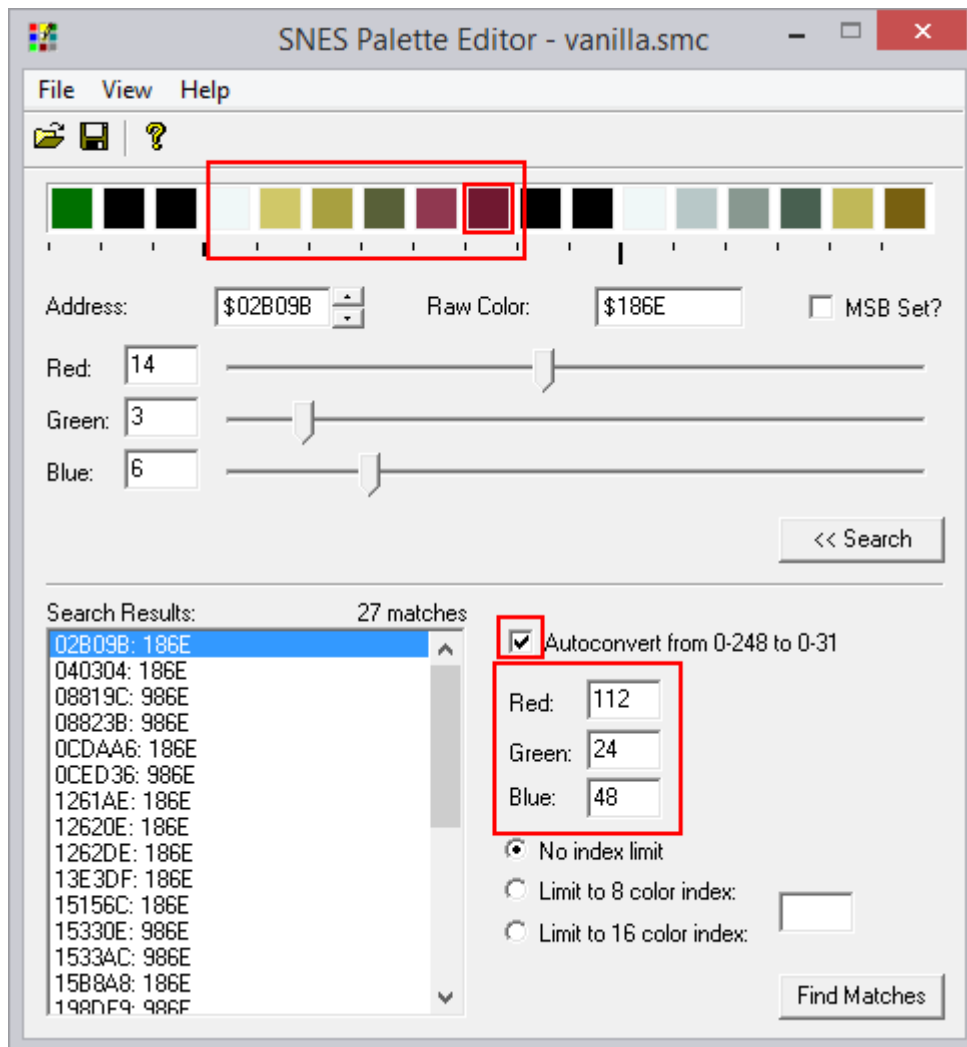
You can right away edit your new palettes. It can be unclear how many exclusive colors your GFX has but try reusing the available white and black on the palette and only edit what seems to be exclusive. You minimize this way the potential conflicts. Gimp Colormap for indexed images is useful to get the right RGB values to edit in SNESpal, however you need to do an hexadecimal to 1-31 scale conversion by dividing by 8.
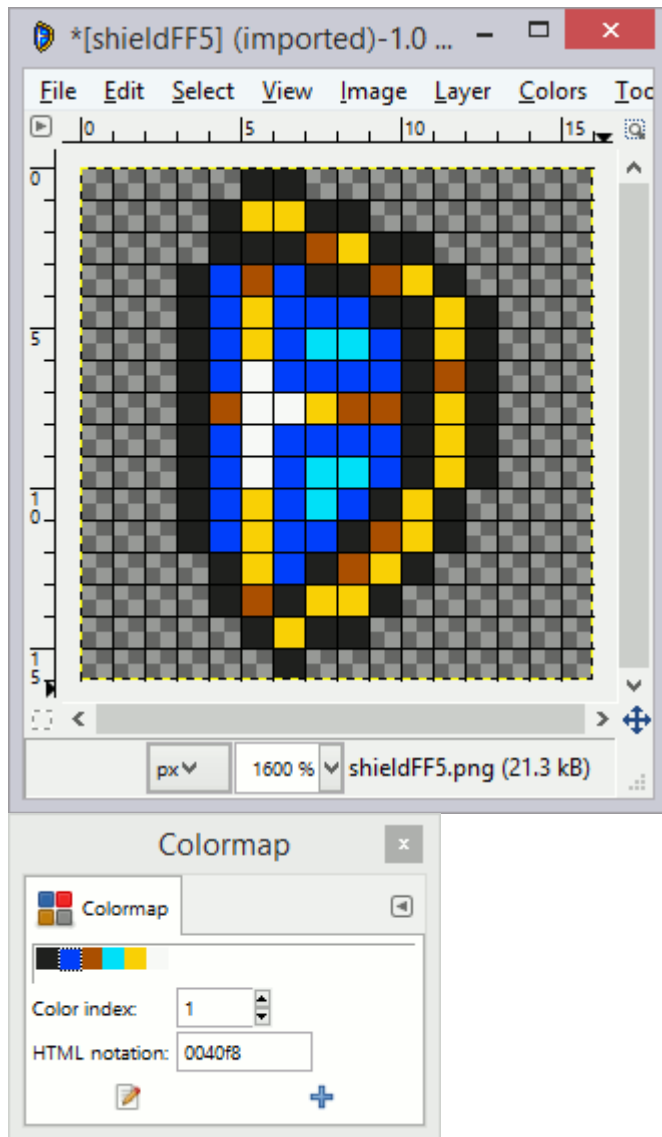
Editing the Graphic

This is the fun part! (to me..)

First you can edit your YY-CHR palette to what was in Gimp and SNESpal, as shown here. Next just edit the GFX and the finalk result should look like this:
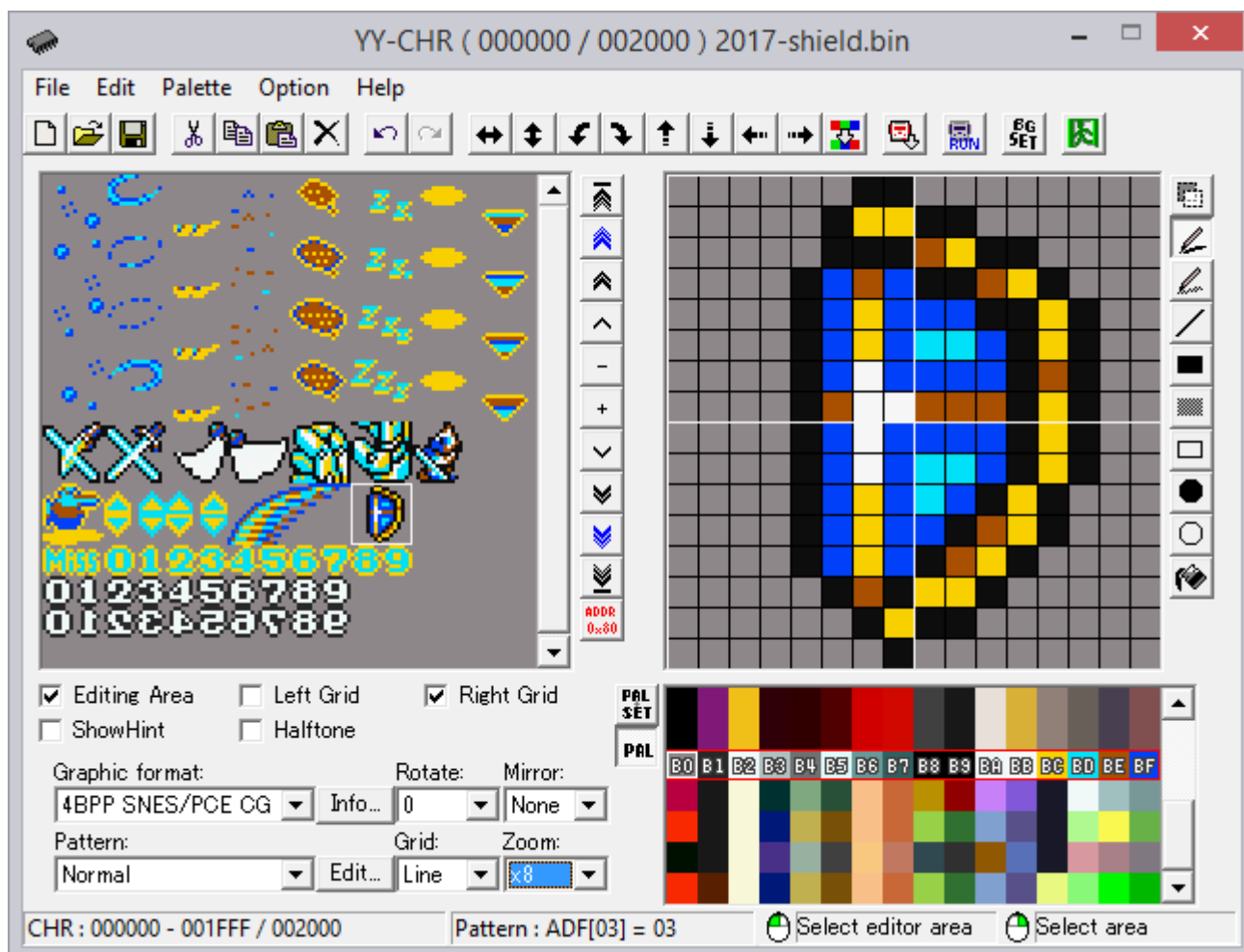
Recompressing the Graphic

You can now reload your ROM in PSV and enter the Right info as if you were decompressing. You should in this case enter $D2EBFF as ending offset to have some room if your GFX take more space (I have not validated if this really has an impact or not). Click once again **Compress** and enter your starting offset in the compression window. Click finally **Recompress** and save your ROM.
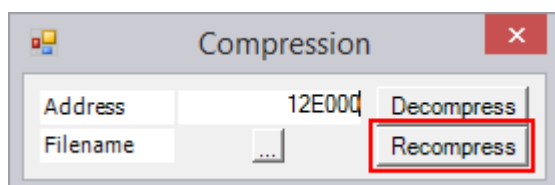
The result in-game should be this one:

Conclusion

I hope I made this tutoriasl clear enough to give you good indication on how to proceed about editing compressed graphics. If you ever got a compression error due to lack of space, there is no simple trick: you either relocate the GFX which require code modifications or make your graphic take less space once compressed.

Video

From:
https://www.ff6hacking.com/wiki/ - **ff6hacking.com wiki**

Permanent link:
https://www.ff6hacking.com/wiki/doku.php?id=ff3:ff3us:tutorial:compressed&rev=1505351073

Last update: **2019/02/12 10:52**