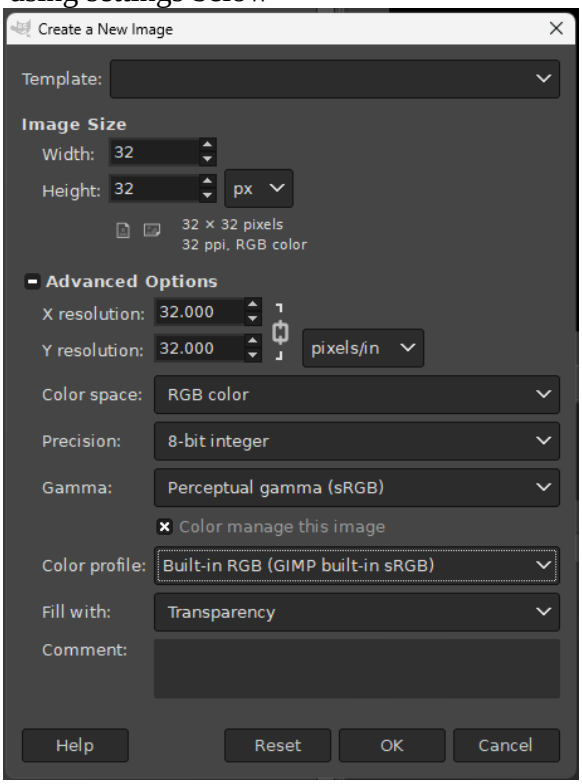
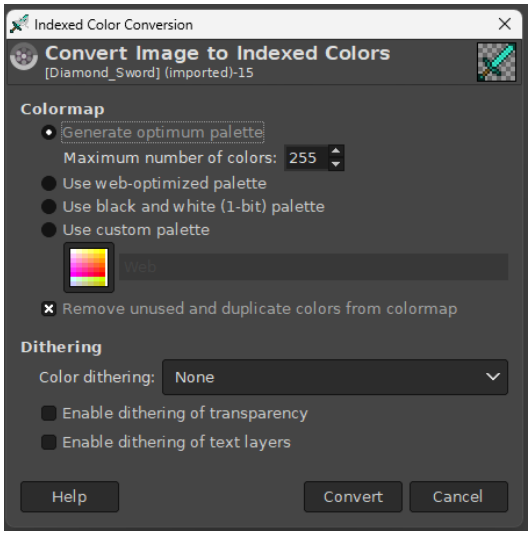
For starters…
I am not an expert… at any of this… not the programming, not the reverse engineering of anything, and certainly not pictures and their file structures. Everything documented here is just an account of my experiences through trial and error, and with the guidance of some very helpful tools (credits listed at the end).

What works? (I used **GIMP** and **Pixelformer** for all graphics testing, below includes the settings I used for GIMP)

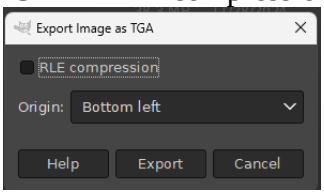
- Walkthrough for creating an applicable item icon using GIMP
  - Create new image using settings below



  - Draw or Copy/Paste your image
  - Image → Mode → Indexed…



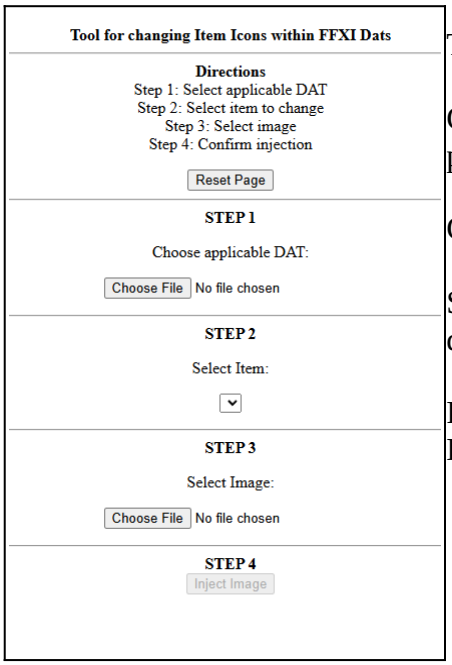
- Image →Precision → 8 bit integer & Perceptual Gamma (sRGB)
- Image → Scale Image → Width & Height: 32 px, X res & Y res: 32 px/in, Interpolation: None
- Export As… → ‘filename’.TGA → RLE compression UNCHECKED; Origin: Bottom left



The basics…
“Item icon images” used in the DATs are 256 color indexed bitmaps, and it appears as though the original file format SE used was TGA. This is an assumption due to the way the color palette and raster data are organized in the DAT. The 256 color palette (RGB) has an alpha channel (A), so with 4x channels and 256 colors the total size of the color palette is 1024 bytes. In the TGA file structure the raster data immediately follows the color palette, which perfectly aligns with the image data found in the DATs. This makes injecting the TGAs easier.

I am working on extending the functionality of the tool to include other image formats, such as BMP. More to follow on that. BMPs are a possible filetype to use, but with caveats. BMPs don’t save the alpha data in the color palette, so typically the entire alpha channel will save as 0x00 as opposed to 0xFF, or whatever the alpha value is for that RGB color on the palette.

How to use this repo…
This is all Javascript frontend, and relies on accessing local files. I had been working on other projects at the time this idea came to me, and I just kept working in Javascript for a solution. In order to run the HTML file correctly you need to have a Cross Origin Request addon for your browser to allow the JS file to load, and download/clone the repo, open in Visual Studio, and ‘Go Live’.



The page sets up a series of 4 steps.

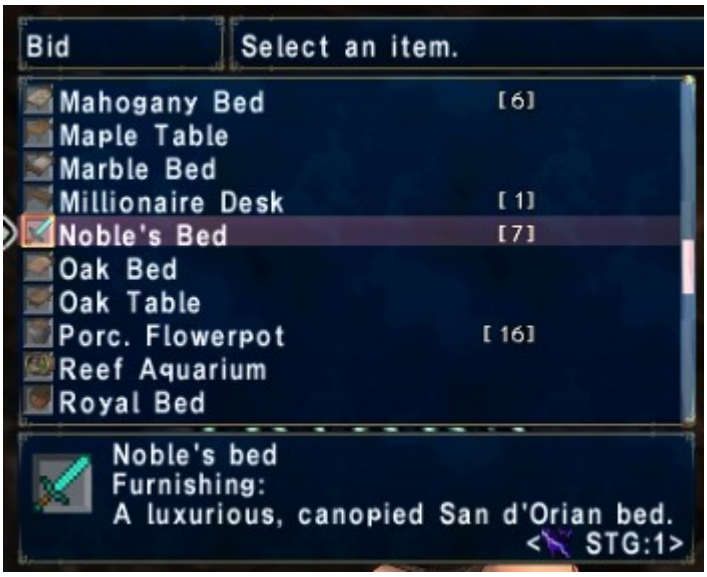
Choose the DAT to inject. Once you choose the DAT, Step 2 items populate.

Choose the item to overwrite.

Select the new image. For now this must be a TGA file, with the details listed above.

Finally, injecting the image will cause the program to encode a new DAT file and download to your downloads folder.

An example of the resulting DAT in game. For this example I replaced the icon for ‘Noble’s Bed’ to be a Diamond Sword from Minecraft. The swap took a total of 5 mins, including finding and downloading the original minecraft image.



Credits:
@Hugin – for the idea !

Shining Fantasia
https://github.com/clanofartisans/ShiningFantasia

Nomidos FFXI Labor
https://www.nomido.at/ff11/

POLUtils
https://github.com/Windower/POLUtils

Windower’s Addon Equipviewer (Rubenator)
https://github.com/Windower/Lua/tree/live/addons/equipviewer