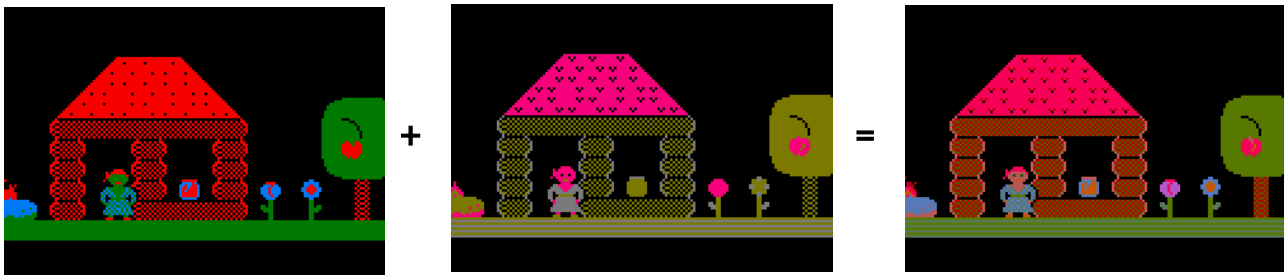
Some techniques used in „Paulina's Potions"

Color flipping

...was invented in the Demo Scene, first and foremost by the group „Les Sucres à Morceaux" with their „Climax"-Demo. To my knowledge, it has not yet been used for a game (if I'm wrong, I'm happy to learn, if anyone knows more...) The principle is very well explained on http://cpc.sylvestre.org (In french, merci beaucoup, Sylvestre!). Every frame, you flip two slightly different images with quite different color palettes. You get a more or less flickering image with theoretically up to the square amount of colors. In mode 1, that's 16.



After some experimenting, I settled for black in the background (in both palettes obviously) and 6 main colors with similar luminescence for less flickering. I avoided to flip the colors with black, so I ended up with $3^2=9$ colors. Testing them on a real CRT. three of them were really nice without flicker, two were rather bad and four were acceptable.
Technically, I just set up a double buffering system, but with different palettes for each screen and of course all the tiles and sprites had to be stored in both versions, so double the memory use, but less memory because two screens. That's why I tried to keep it simple, f.i. the potion symbols are stored monochrome like the font, color is applied on the fly.

Screen width of 86 bytes (43 characters)

In Druid & Droid, I used a 64 byte-wide screen to make **fast drawing routines** (*inc l* instead *inc hl*) possible. With the 86 byte width in this game, I have the same advantage, because all the hibyte-increases happen to be on the right 20 bytes of the area. This way, I can have a nice sidebar menu AND fast sprite drawing.

Tilemaps

That sums up to stages of 32x20=640 tiles. I developed a compression format to reduce this size to <256 / stage and a very ugly C-program (not in the sources!) that compresses the .tmx files made by Tiled and writes an .asm file to include in the main program.

Optimizing

is a really unhealthy obsession of mine. Self modifying code, strange algorithms and unnecessarily complicated byte-saving formats for the enemy and item lists, unreadable spaghetti code. And ironically, in the end, I even had over 4K unused memory left. ¯_(ツ)_/¯

PS.: Spoiler alert: If you didn't notice it already: The walkthrough is printed in magic ink under the paragraph „The Solutions" in the manual