Druid & Droid

...not really a "making of", but some thoughts

What I wanted to achieve:

I wanted to make a game that wouldn't look and feel like the "average" CPC game. I went for Mode 1, but with some unusual color schemes. Instead of the blocky default 8×8 characters, I used my own antialiased, variable-width font. I wanted to avoid slow double-buffering and lagging screen movements, although that meant less and smaller sprites. I wanted to have several tunes for the different missions, but all based on the same musical idea (here comes my background as classically trained musician, which is my day job). I wanted to include level titles, dialogues and animated cutscenes (prologue, sequences between missions and epilogue). And last but not least, I wanted to find a nice story for the game (possibly without violence, laser guns or spaceships) and a concept that involves both quick reflexes and some brain work as well.

Why machine code?

Because I want to feel every single byte. And because I'm just too dumb for higher level languages. I'm not a professional programmer, I have no idea how dev-toolchains and libraries work. I started coding at the age of 12 with Locomotive Basic and couldn't really get away from spaghetti code principles since... Also, I had this dream through my whole adolescence to really master the Z80's native language and write an awesome game for the CPC. But back then, without internet and emulators, it found it really hard to develop my skills. Nowadays, with a certain level of immersion, it was actually quite easy. And actually a lot more fun than my attempts to learn C++ or Python. Now, coding for CPC to me is like playing with model railways for others. I'm sure one could write the same game in C with a lot less work and way more beautiful code, but honestly, *I* couldn't.

A matter of taste: Self Modifying Code

In my opinion: Yes. I love it and I'm afraid most of my coding wouldn't work without it. But I'm not a professional. Mainly for speed reasons, a lot of my variables are in-line. I re-use subroutines for modified purposes (f.i. drawing and then erasing) by just changing a handful of bytes directly in the code to avoid redundant code or excessive branching. Most of the different behaviour in some missions (swapped command in mission 6, bouncing in mission 7) is done by directly injecting different values and opcodes into the code.

Memory optimization:

The 64K rule that everything has to be loaded at once at startup led me to some strange design choices and to manically over-optimizing my code from the beginning. Retrospectively, not all of it would have been necessary, but I couldn't estimate the size of the project when I started it.

Some of the optimization turned out nicely: I could fit all eight background images in quite a small space by using a set of 68 16×16 meta-tiles that consist of combinations of 225 8×8 monochrome sub-tiles. Likewise, the large font is saved just monochrome (and compressed) and antialiasing as well as color effects are applied on the fly. I'd like to think my level data compression is reasonably simple but efficient. In the music as well, I could afford to have a different tune every mission by using a lot of repeated and transposed patterns to keep the data small. I also squeezed data in places where it doesn't belong but where I found empty space, f.i. where the tiles of the last mission (epilogue) would be stored.

In other parts, I really got bogged up with my own optimization ideas: The worst were the text shortcuts, itself a nice thing, because there's a lot of text in this game. But I went to far, switching the places of characters and leaving the ASCII order just to gain a few more bytes. That lead to completely unreadable text strings in the code and turned adding or changing text to a nightmare. And then the decision to use a mapping for the level tiles where each type of tile has their own bit… Yes, that's useful for checking possible movements and collision with enemies, but of course it reduced the total amount of tiles greatly. I could have used a lot more and diverse tiles (also some with additional functionality, like switches or doors) if I'd done it differently. For now, the different platform and obstacle variants are in fact applied by random.

And then there were some quirky ideas where I don't know if they make sense ore are just a spleen, like not having a proper variable for the input, sound and monitor settings but instead xor'ing the text with a string that turns "MONO " to "COLOR" and then just reading the first character to decide what palette to use…

Summary:

I spent very much time on this game. Too much, if you ask my family. There would have been easier ways to achieve this, but I'm really happy with the result and how I got there. Making this was a great experience.

Leonard Eröd *alias* arnOLdE