The Making of Fips‘ Tale

Background

When I was a child, my dad introduced me to video games. I remember especially the first ones, most of which were on the Amstrad CPC – the likes of Elite, Tau Ceti, Antiriad, Boulder Dash, Prince of Persia, , and many others that I have fond memories of. We also had a physical CPC 664 machine at home so I was able to get some kind of second-hand retro experience.

I also learned different programming languages over the years but had never thought of making a game on a retro system.

The Beginning

My dad, who‘s keeping up with the CPC scene, told me about the competition in mid July 2020. At that time I had no other game project going already so I decided to try and submit a game (first commit in my Git repository was on July 23). So I went and learned C, and also started using CPCTelera, conducting basic tests for drawing sprites and tilemaps while coming up with a game idea. I made some initial sketches that are still quite close to the finished game.

Initially, progress was slow because I had to get accustomed to C and the CPC. I learned that you shouldn‘t be lazy with debugging tools. It got much better after I started to use the debugger of WinAPE and wrote my own text rendering function (needed if you overwrite the builtin Amstrad font), which sped up debugging quite a bit.

The CPCTelera examples and demo projects also helped quite a lot and gave me an idea of what a decent project structure might look like.

Development

I started out with the concept of a green, grassy platform game where you‘d collect coins and keys which would be used to extend bridges in order to progress while fighting the ocasional enemy in a Prince of Persia style sword fight.

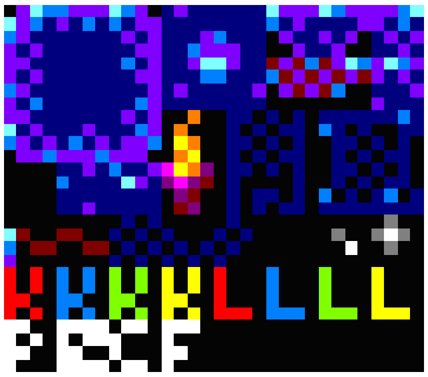
Image 1: The tileset

Some weeks into development it became clear that going for a nature setting was more challenging artistically and, also given the limited palette of the CPC, I switched to a darker setting inside a castle, which also helps to set the mood (and I'm not a great artist either).

I was able to work faster over time as I was getting used to how things worked. I also learned about optimizations for loops and other things like inlining small functions in tight loops (mainly thanks to „Professor Retroman" Videos).

The AI in the game got roughly three iterations. In the current one, it's both acting randomly and reacting to what the player does. It attacks, defends, steps backward (without falling off ledges) and pursues the player once it has been alerted, again avoiding falling from a ledge to death.

I admit that I had a look at the source code of Prince of Persia (here on GitHub, Apple II version) every now and then, but that out of interest and for inspiration, not for technical details.

The screen system

I quickly realized my levels would need large amounts of memory, so I looked into compressing individual tilemaps (called screens here).

One screen has 40x40 tiles, so that's 1600 bytes per screen. A level has a maxiumum size of 3x3 screens which could use 9*1600 = 14,4kB for just one level!

When using the zx7b compression included with CPCTelera 1.5 on each individual tilemap, each would only use between 50-200 bytes which oviously saved lots of memory.

Finishing up

The last four (or so) weeks I was in a bit of a rush. I put in the last features and designed out levels while also looking to create a casette loading screen, main menu etc. At that time I also included player routines from Arkos Tracker 2 for music and sound effects (which turned out alright given my mediocre music skills).

That mainly worked but I couldn't polish all of it to a level that I was completely happy with. On the other hand, that's the thrill of a game jam, isn't it? Learning to work faster with time constraints.

For me the best way to design levels was to sketch their rough structure out on paper, then block them out in Tiled, after that put them into the game and playtest them. The last step was decorating and making minor adjustments.

Initially a two player mode was planned, where you would have a sword fight against each other and some kind of point system for each victory. However, time ran out, altough the code is designed to work with any kind of input, be it an AI or a human player. I might add that in a post-competition version of the game (there also are some minor bugs still lingering…)

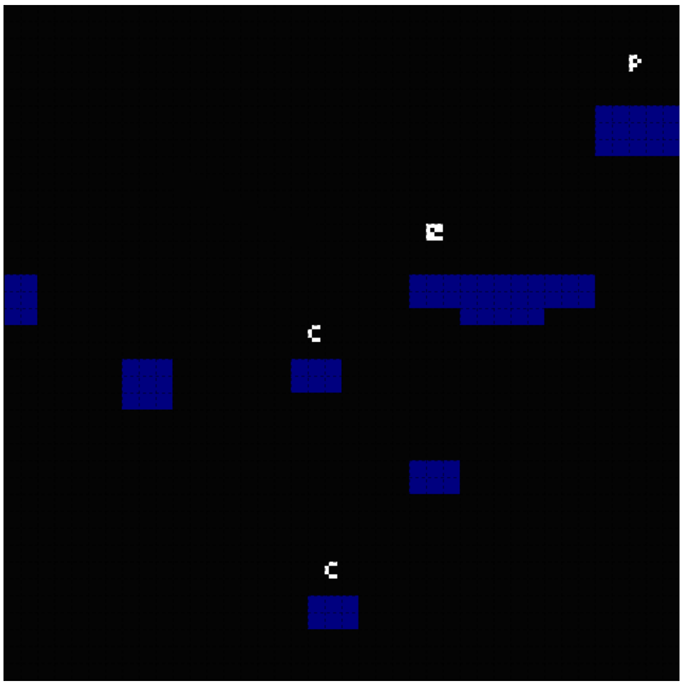
Image 2: Blocking out levels in Tiled

Learned Lessons

For my next game of this type, I'd probably use 8x8 tiles (saves memory and can help graphical fidelity), make bigger sprites and use Mode 1 if I don't need that many colors anyway.

I'd also make a more torough entity system that'd be more flexible than my current approach. This would also involve animations, collectables etc. Currently an entity component system (ECS) looks favorable and may even be benefical for performance.

We'll see what happens in the next years so… stay tuned :)

In summary I learned a lot, got lots of help from the nice community on the cpcwiki.eu forum and generally had a lot of fun building Fips' Tale.