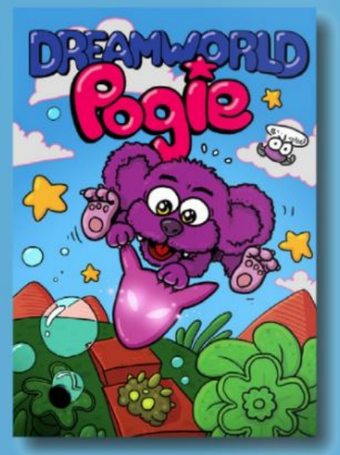
DREAMWORLD Pogie



Dev Diary

"Hello, I'm Lyndon Sharp, Lead Programmer at W.A.S.P (We are Spectrum Programmers)"

This all started when I became aware of a new "super" Spectrum. Initially I approached my old mate Philip Oliver about doing Dizzy for the Next but they had that covered already so they offered me *Pogie*, which was a decent platform game on the Nintendo NES. Now I was still working for the Oliver Twins while Andrew Oliver was writing it so I was quite familiar with it already.

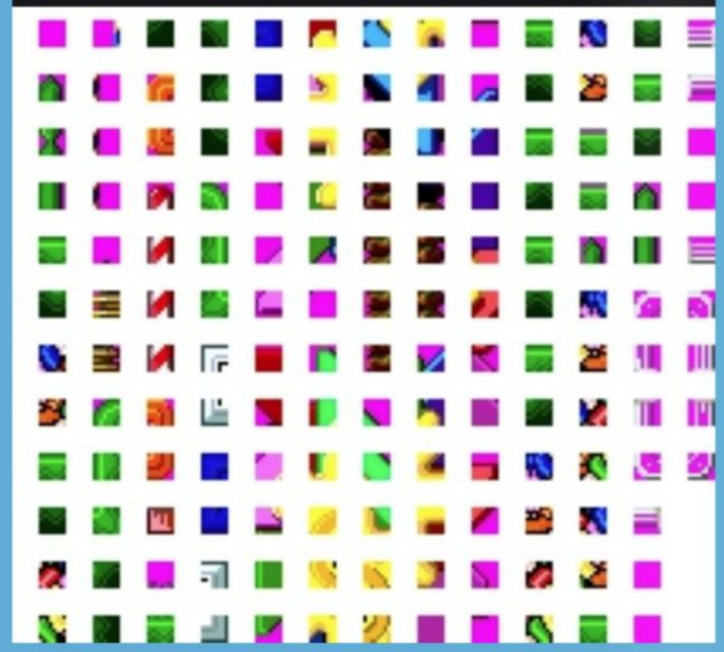
Next job was to pledge for a Next development board on Kickstarter, but before I even got that far I had a word with Henrique (or H as I like to call him) and before I knew it I had a free board, an issue 1, on my desk ready to play with. So off we go…..

Well not quite, my original artist just didn't have time so I had to put the feelers out for a new artist. This is when Phoebus Dokos threw his hat in the ring and together we reformed WASP.

I had quite a lot of resources given to me by Philip including bitmaps of the original levels. This was really handy as it enabled me to be able to write a tile cutting tool on the PC while Phoebus was re-colouring the first level in full 256-colour splendour. Then, after 12 months development of the tool chain and the game itself, I was struck down with leukaemia for the second time which halted development for the next three years. Still we are back on track now.

Next came the Scrolling Routine, this took quite a while as it was about 50 times more complicated than I could have ever imagined. It needed to be able to run any pixel speed from one to eight and be able to change speed or direction at any point, to be fast and be able to work seamlessly with 16K memory paging for both level data and tile data (this was way before tile mode existed). I even needed a custom Z80n instruction LDWS which makes vertical tile data drawing to *Layer 2* very fast.

Put all these together, and it would make great bedroom wallpaper!



My tile chopper works by chopping out 8x8 pixel blocks from a .bmp file and storing them in a buffer, any further cut blocks get checked against this buffer. If it finds a duplication it just puts this index in the map data file otherwise it adds the new tile to the buffer and puts that index in the map data. Map data is 16-bit, 12-bits of tile indices (4096 tiles) and 4 bits of attributes like solid, semi-solid, sprite markers or instant death etc. It also saves out a test screen of all blocks so it can be visually checked which helps spot similar graphics that are only one pixel different.

These tiles get reconstructed into the level by the scrolling routine.

The first stage of the Scroll system builds a set of tile indices to be used on the stack. The second stage pulls these tile addresses off the stack and renders the vertical pixel stripes using the LDWS instruction I mentioned earlier. The purple colour represents transparency so anything under the *Layer 2* will show through here. In Pogie this is a scrolling collection of variable speed clouds and some ¼ speed pyramids and a sand scape parallaxed in *Layer 1* using the raster interrupt.

This is just a wrapping set of graphics as having two scrolling routines would have taken valuable "T" states away from other systems. It gets the job done quite nicely and it's virtually free with regards to processor time.

Next time I will talk about sprites, how they are used, constructed and some tips and tricks we used in combination with exploiting the DMA.

Putting the bits together, and soon the level takes shape.

Pogie bouncing around a very colourful, parallax world!