Spectrum Analyser

Instructions

Introduction

Spectrum Analyser is a tool that is designed to help reverse engineer ZX Spectrum games & programs.
It is a combination of an emulator, debugger & interactive disassembler. Programs are loaded and run in the emulator and their state can be inspected using the tools provided. These tools include:

- Disassembler
- Debugger
- Graphics viewer
- Frame trace
- Memory diff
- IO analyser

The analyser maintains a representation of the program which is the results of analysing what is run as well as comments and other input by the user. This representation can be exported as a .skool file to be used with skoolkit (www.skoolkit.ca) or an assembler file.

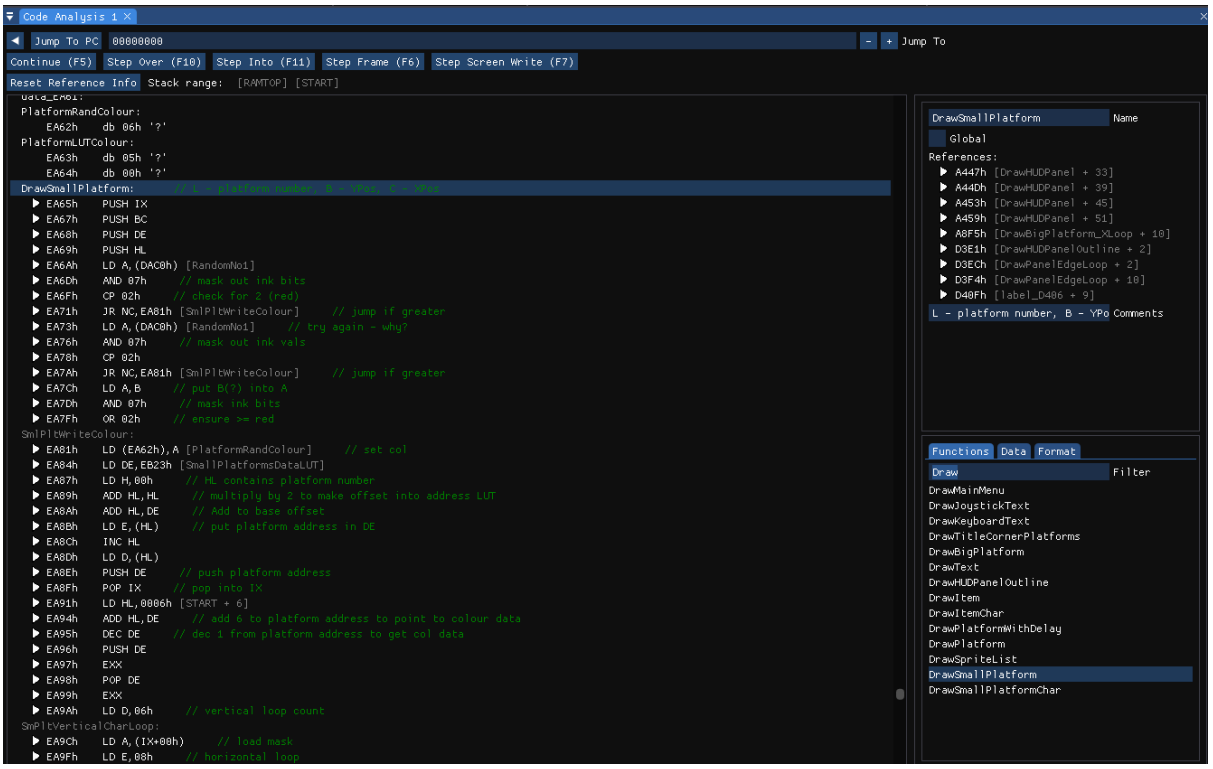
Acknowledgements

This program was built using the superb Chips emulator library by Andre Weissflog, the emulation in the analyser is done using this library: https://github.com/floooh/chips

For the UI I used DearImGui (https://github.com/ocornut/imgui) which is without doubt one of the greatest pieces of open source software. Without it not would the UI much longer but working on it would also be exceedingly dull.

Code Analysis Window

This window is where the bulk of the analysis takes place.



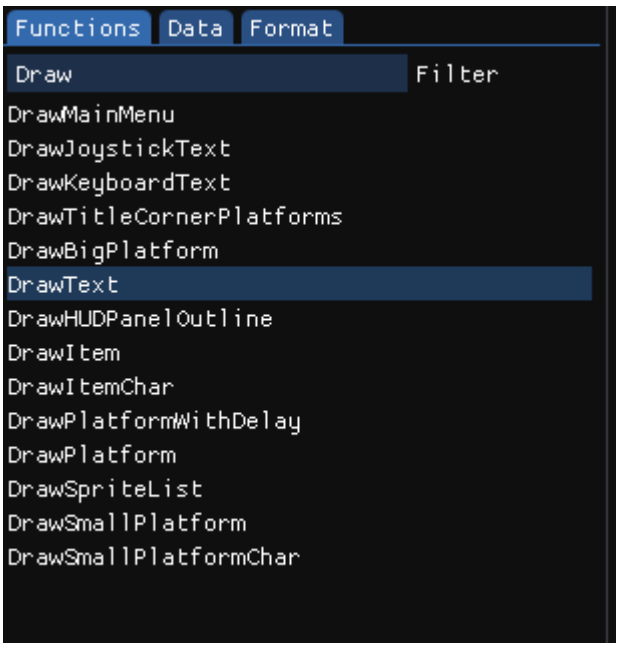
Main View

The main view covers that address range of the ZX Spectrum and has both code and data items.

Detail View

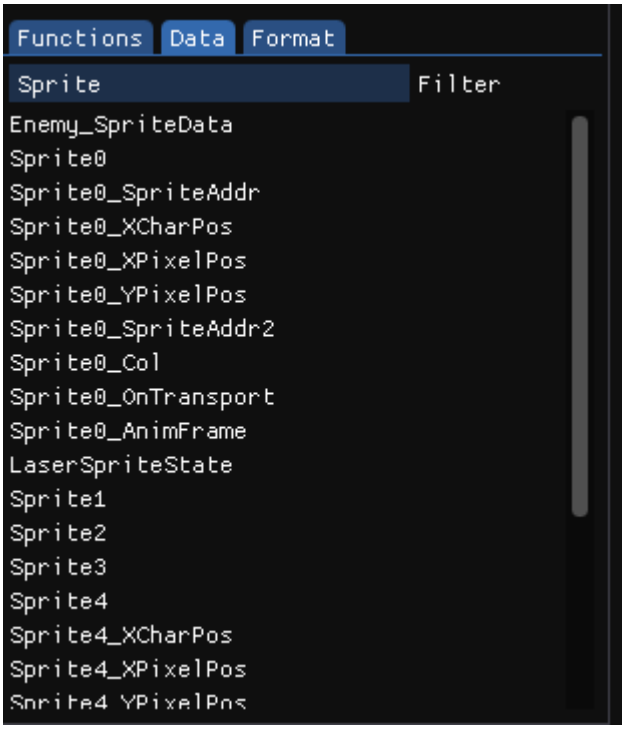
The details view will show extra information on the selected location and is the pane in the top right of the above screenshot.
Information about what is accessing the selected item can be found here.
For data items you can see what is reading and writing the value as well as how the value is changed.
For code items you can pick the operand type to customise how the disassembly is displayed. You can also ‘NOP’ out instructions so see how the behaviour of the program changes when the instruction is removed..

Functions



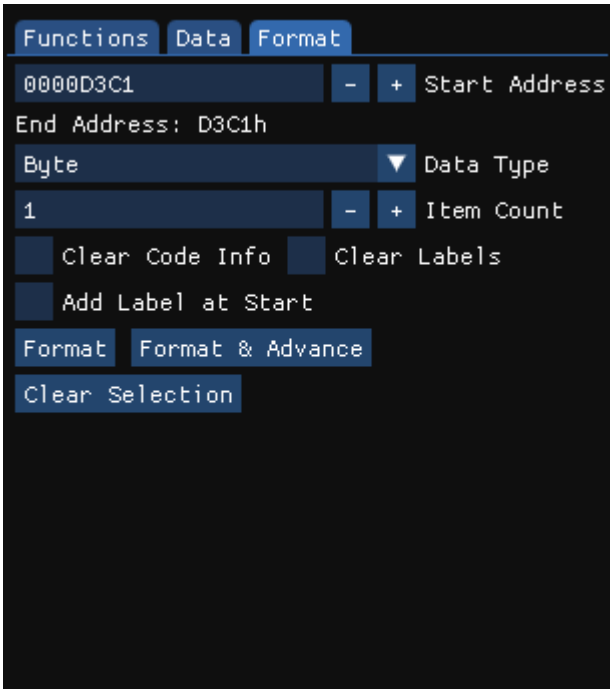
This lists all the globally declared functions and the use can jump to them by clicking on them. They can also filter the list.

Data



Similar to the function list, this is where all the globally declared data is listed and it can be filtered in the same way.

Format



This is a tool which allows the user to format an area of memory in terms of how it is visualised.
‘Start Address’ is the starting point of where you want to format.
The data type is the type to format. You can then set the number of items of that type.
You can clear the region of labels or code if you want (by default the operation won’t override code)
The ‘Format’ button will then format that data region. ‘Format & Advance’ will format a region and then move the start address to after the formatted region, this is handy for formatting consecutive items in memory.
You can also add labels to the start of each item by using the ‘Add Label at Start’ check box.

Keys

Debugger

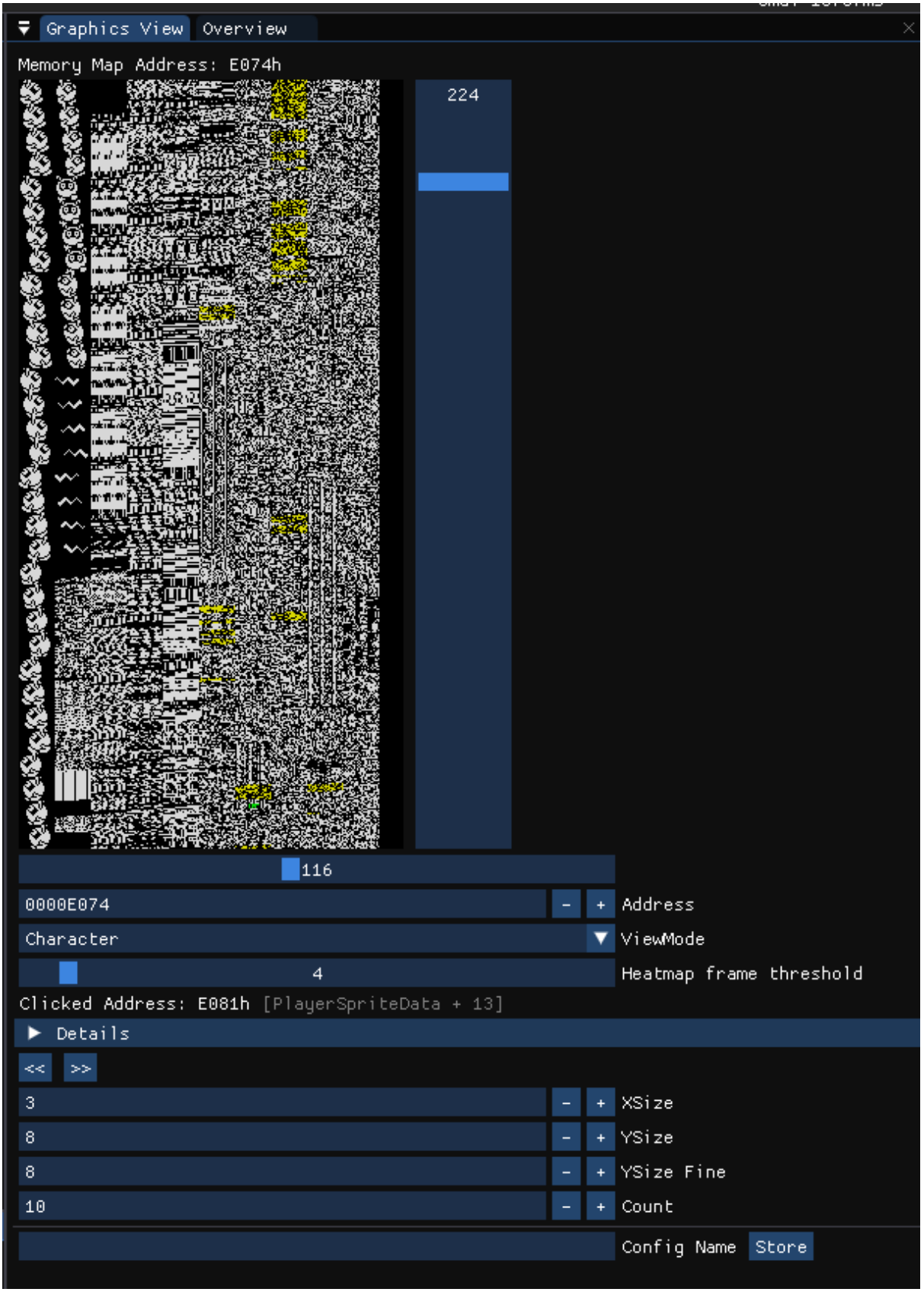
- F5 - Break/Continue
- F10 - Step Over
- F11 - Step Into
- F6 - Step Frame
    - This will make the program run for a frame
- F7 - Step Screen Write
    - This will make the program run until the next screen write operation
- F9 - Add/Remove breakpoint

Program Annotation

- D - Set item data
    - Toggles between db & dw
- T - Set Item text

- C - Set item code
- B - Toggle item binary representation
- L - add label
- R - rename

Graphics Viewer



This view is used to find the graphics in a game. It draws the memory graphically using various visualisation modes:

- Character
    - A column based character view. Most games seem to store their sprite and other graphics this way. The character width can be adjusted for graphics which have a width of 16,24,32 etc.
- Character Winding
    - Some games seem to store their characters in a left->right->left winding pattern.
- Screen
    - This is will draw memory in the same way the screen is drawn

Frame Trace

When the program is running the state of the last 300 frames is recorded. At any point you can scrub across this timeline and for each frame get an instruction trace as well as see all the screen writes.
Execution can be resumed from a previous frame. Memory between frames can also be diffed.