State of Retro Gaming in Emacs

Vasilij Schneidermann

November 2020

Outline

Section 1

Intro

About

- Vasilij Schneidermann, 28
- Cyber security consultant at msg systems
- mail@vasilij.de
- https://depp.brause.cc/
- http://emacsninja.com/

Motivation

- Emacs is the ultimate procrastination machine
- Many fun demonstrations:
  - Order salad online
  - Window manager
  - IRC bot
  - Textual web browser
  - Basic games
  - 3D maze
  - Z-Machine emulator
  - Audio/video editor
  - Sex toy controller
- Can we emulate retro games at 60 FPS?

Meet chip8.el

- https://depp.brause.cc/chip8.el
- Pretty much finished, <1000SLOC
- Supports Super CHIP-8 extensions
- Runs at full speed, games behave OK

Section 2

Fun facts about `chip8.el`

What the hell is a CHIP-8 anyway?

- It's a VM, not a console
- Designed for easy porting of home computer games
- Not terribly successful
- Small community of enthusiasts writing games for it
- There are even a few demos!

System specs

- CPU: 8-Bit, 16 registers, 36 fixed-size instructions
- RAM: 4KB
- Stack: 16 return addresses
- Resolution: 64 x 32 black/white pixels
- Rendering: Sprites are drawn in XOR mode
- Sound: Monotone buzzer
- Input: Hexadecimal keypad

How does it work?

- Runs at an unspecified speed
- Sound and delay timer count down at 60FPS
- Game is loaded up at #x200 into RAM
- Program counter is set to #x200
- Decode instruction, execute, loop

Game loop woes

- Game approach: Do stuff, wait, repeat
- Doesn't work well in Emacs due to user input
- Interruptible sleep: Unpredictable
- Un-interruptable sleep: Freezes
- Timers: Inversion of control, allows user input to happen
- Call a timer function at 60FPS, don't do too much in it:
    - Execute CPU cycle(s)
    - Decrement sound/delay registers
    - Repaint

Mapping the system to Emacs Lisp

- It's all integers and vectors (of integers)
- RAM, registers, return stack, key state, screen, etc.
- Stored in global variables
- No lists are used at all
- Side effect: No consing happens, no GC pauses

Decoding instructions

- All instructions are two bytes
- Arguments are encoded inside them
- `JP nnn` for example maps to `#x1nnn`
- Type extracted by masking with `#xF000`, then shifting by 12 bits
- Argument by masking with `#x0FFF` (no shift needed)
- Common patterns emerge, like addresses being the last three nibbles
- Big `cond` dispatching on the type and executing side effects

Testing

- Initially: Execute ROM until user hits `C-g`
- Use debug command to render screen to a buffer
- Initial test with tiny ROMs that just display a static screen
- I added instructions as needed, went through more of them
- Later I wrote a unit test suite as safety net
- Each test initializes the VM, loads up code, executes the `chip8-cycle` function, checks for side effects

Debugging

- My usual approach of using edebug was ineffective
- Therefore: Logging it is
- I compared my log output with an instrumented version of this emulator: `https://git.foldling.org/chick-8.git`
- If the logs diverge, that's where the bug lies
- Future project idea: A CHIP-8 debugger

Analysis

- Writing a disassembler is simple, but tedious
- Adding analysis functionality is particularly tricky
- Idea: Reuse radare2 framework, add analysis/disasm plugin
- I wrote one in Python, then discovered there is one in core. . .
- I then improved that one to the same level

Rendering

- By far the trickiest part
- I intentionally decided against using a library
- Creating SVGs: Too expensive
- Creating/mutating strings: Too expensive or complicated
- Changing SVG tiles: Gaps between lines
- Bool vector backed XPM: Caching effects ruin everything
- Plain text with background color: Perfect
- Many optimization attempts until I got there

Sound

- You only need a beep, so no difficulties emulating it
- Playing it is hard because Emacs only supports synchronous playback. . .
- Emacs processes are asynchronous, so controlling one works
- `mplayer` has a slave mode, `mpv` supports listening on a FIFO for commands
- Proof of concept:
  - Start paused `mpv` with a FIFO in loop mode
  - Send pause/unpause command to the FIFO

Section 3

Outro

What next?

- Maybe an Intel 8080 emulator running CP/M
- Maybe experimentation with faster rendering
- More serious stuff in CHICKEN, like NES or GB emulator

Questions?