Emulating a Sega Master System in Javascript

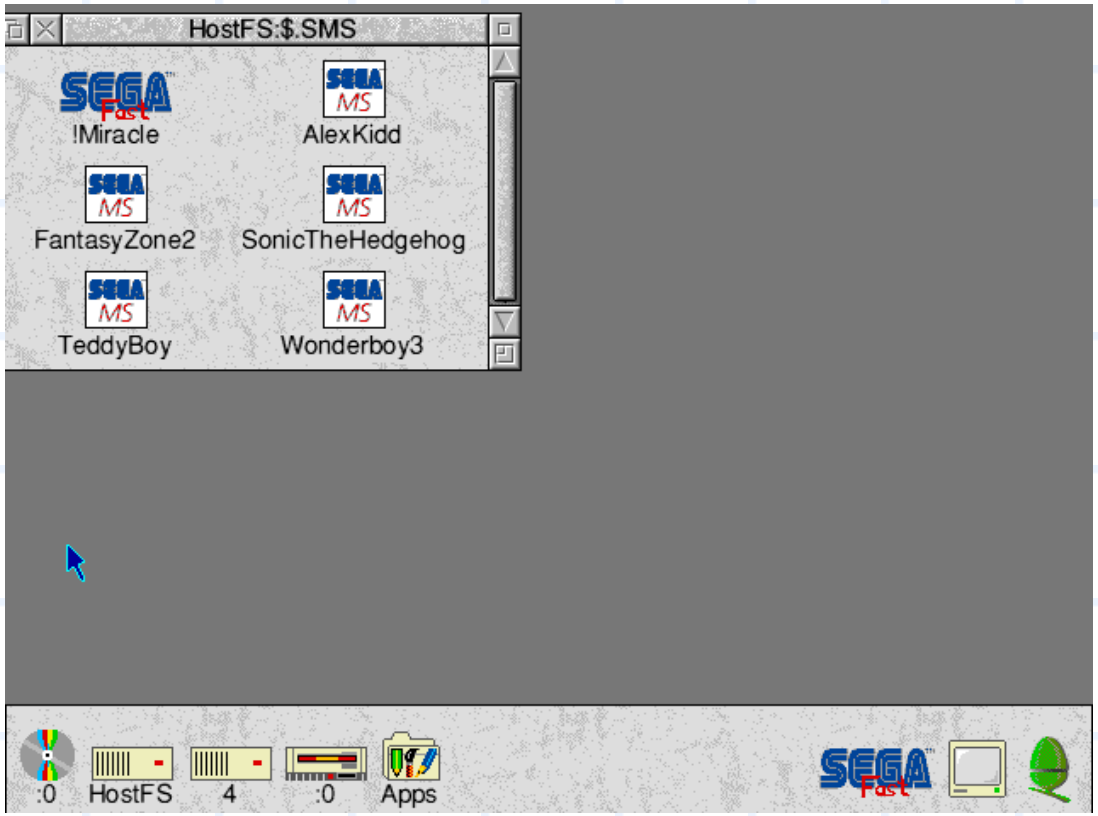
Matt Godbolt

http://github.com/mattgodbolt/Miracle

Why?

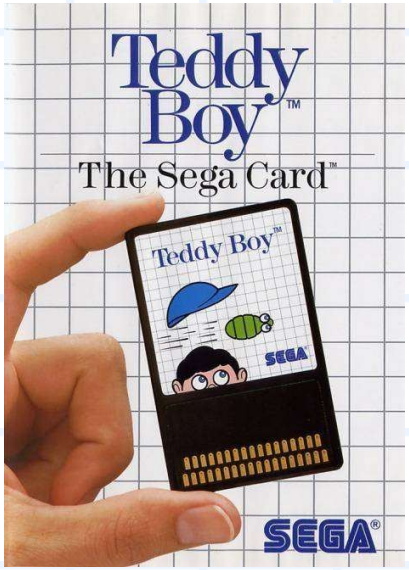
Back story

Back story

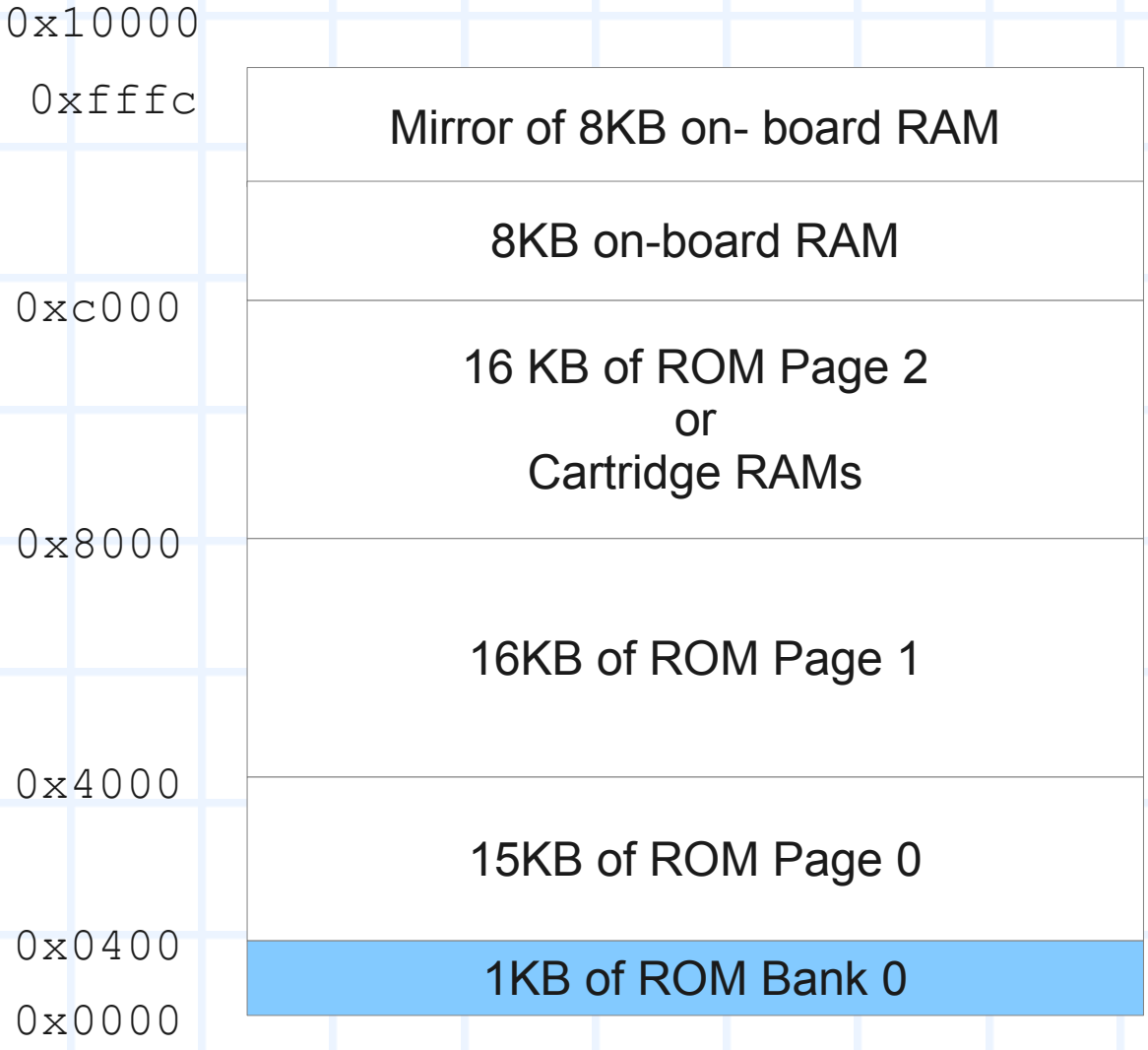

HostFS:$.SMS
!Miracle
AlexKidd
FantasyZone2
SonicTheHedgehog
TeddyBoy
Wonderboy3
:0
HostFS
4
:0
Apps

What's inside

- 8-bit Z80 CPU
- 8KB RAM
- Custom VDP
  - 16KB RAM
  - 256x192, 64-color
- SN76489 Sound Chip
- 32, 64, 128, 256 KB ROMs



Memory Map




```
0xffff : Page 2 RAM
0xfffe : Page 1 RAM
0xfffd : Page 0 RAM
0xfffc : ROM/RAM sel
```

```
NMI/IRQ handlers
"RST" handlers
```

Memory Map

```
function readbyte(address) {
  if (address < 0x0400)
    return romBanks[0][address];
  if (address < 0x4000)
    return romBanks[pages[0]][address];
  if (address < 0x8000)
    return romBanks[pages[1]][address - 0x4000];
  if (address < 0xc000) {/*ROM page 2 / cartridge ram*/}
  if (address < 0xe000) { return ram[address - 0xc000]; }
  return ram[address - 0xe000];
}
```

Z80

- CISC chip
  - 900 instructions
  - no multiply or divide
- Separate I/O bus
  - peripherals e.g. VDP, sound
- 3.53 MHz (approx)
  - 4 cycles/1us minimum
  - 7 cycles for LD/ADD
- 18 8-bit registers
  - A, B, C, D, E, H, L
  - A', B', C', D', E', H', L'
  - Flags, IRQs, R register
  - Pairable for 16-bit
    - AF, BC, DE, HL
- 4 16-bit registers
  - Index registers: IX, IY
  - SP, PC

Z80 – Example

```
058a 21 2a 06   LD HL,0x062a    ; HL = 0x062a
058d 87         ADD A,A         ; A = 2*A
058e 5f         LD E,A
058f 16 00      LD D,0x00       ; DE = (u16)A
0591 19         ADD HL,DE       ; HL = HL + DE
0592 7e         LD A,(HL)       ; A = *(u8*)HL;
0593 23         INC HL          ; HL++
0594 66         LD H,(HL)       ; H = *(u8*)HL;
0595 6f         LD L,A          ; L = A
0596 e9         JP HL           ; jump to HL
```

Z80 – Decoding

- Optional prefix (`cb`, `dd`, `ed`, `fd`)
- Opcode byte
- Operand bytes (0, 1 or 2)
- Examples
  - **`XOR`** `A, A` → `af`
  - **`LD`** `A, 0xff` → `3e ff`
  - **`LD`** `B, (IX + 0x0f)` → `dd 46 0f`
  - **`LD`** `DE, (0xd019)` → `ed 5b 19 d0`

Z80 – Executing

```
switch (readbyte(z80.pc++)) {
   case 0xaf: // XOR A, A
      z80.a ^= z80.a;
      break;
   case 0x3e: // LD A, constant
      z80.a = readbyte(z80.pc++);
      break;
```

… but not quite that easy

Z80 – Executing

- Flags?
  - overflow, parity
  - carry, half-carry
  - add/subtract
  - zero, sign
- Interrupts?
- Undocumented opcodes?

Z80 – Executing

- Complex!
- Borrow open source code from JSSpeccy
  - which was based on Speccy (written in C)

Z80 – ADD A, *XX*

```
const sz53_table = [...computed elsewhere...];
const hc_add_table = [0, H, H, H, 0, 0, 0, H];
const oflo_add_table = [0, 0, 0, V, V, 0, 0, 0];
var c = readbyte(z80.pc++);
var result = z80.a + c;
var lookup = ((z80.a & 0x88) >> 3) | ((c & 0x88) >> 2)
           | ((result & 0x88) >> 1);
z80.a = result & 0xff;
z80.f = (result & 0x100 ? C : 0) // Carry
      | hc_add_table[lookup & 0x07] // 1/2 carry
      | oflo_add_table[lookup >> 4] // overflow
      | sz53_table[z80.a];  // sign, zero, "undef" bits
clock += 7;
```

Z80 – Execution

- Code generation from opcode tables:

```
0x00 NOP
0x01 LD BC,nnnn
0x02 LD (BC),A
```

- CPP Preprocessed to Javascript
- Avoids lots of repetitive code
  - 938 instructions
  - 1241 lines of perl
  - 5400 lines of javascript generated

VDP

VDP

- 16KB of RAM
  - accessible by the I/O bus only
- 8x8 4bpp “tiles”
- 2 x 16 entry palettes of 6-bit colour
- Background table
- Sprite table
- Interrupt generator

Tiles

Tiles – Planar mapped

Pixel values	8	0	0	0	0	0	0	15

Bit position	7	6	5	4	3	2	1	0
Byte 1	0	0	0	0	0	0	0	1
Byte 2	0	0	0	0	0	0	0	1
Byte 3	0	0	0	0	0	0	0	1
Byte 4	1	0	0	0	0	0	0	1

- 32 bytes per tile

Background

- 32 x 24 background map of tiles
- Two bytes per tile
  - 9 bits tile index (max 512)
  - 2 bits vertical, horizontal flip
  - 1 bit palette select
  - 1 bit sprite overwrite
  - 3 user bits

Background

Background

20**0212**20202020202020202020202020202020**0616**20202020**474f4c44**20202020

20**0313**20202020202020202020202020202020**07173031**20202020202020202020**34**20

00000000000000000000000**6a7a82928494**00000000000000000000000000000000

00000000000000000000000**6b7b83938595**00000000000000000000000000000000

0000**6a7a82928494**00000000000000000000000000000000000000000000000000000000

0000**6b7b83938595**00000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000**6a7a82928494**00000000

00000000000000000000000000000000000000000000000000**6b7b83938595**00000000

Sprites

- 64 sprites
- 8x8 or 8x16
- 256 byte table sets X, Y and tile index
  - Can only be one of first 256 tiles
  - All use palette 2
  - 64 bytes “spare” … some games pack more tiles in here

Sprites

Sprites

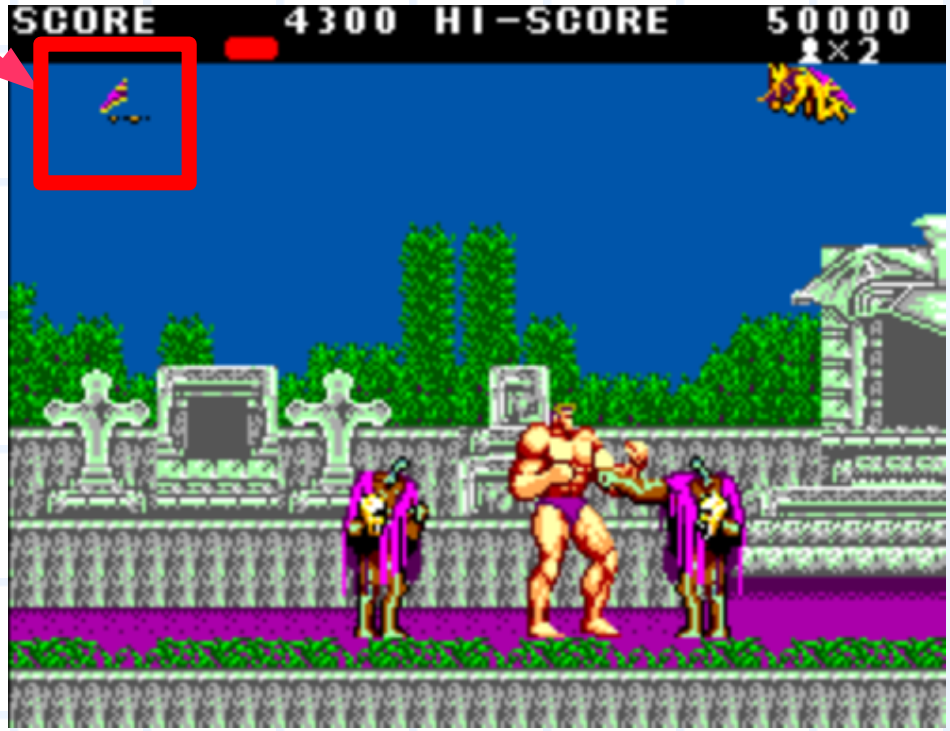
```
0 x: 116 y: 111 t: 16
1 x: 116 y: 127 t: 18
2 x: 124 y: 111 t: 20
3 x: 124 y: 127 t: 22
4 x: 132 y: 111 t: 24
5 x: 132 y: 127 t: 26
6 x: 211 y: 111 t: 102
7 x: 211 y: 127 t: 104
8 x: 219 y: 111 t: 106
9 x: 219 y: 127 t: 108
```

```
10 x: 147 y: 35 t: 44
11 x: 155 y: 35 t: 46
12 x: 155 y: 208 t: 142
13 x: 155 y: 127 t: 144
14 x: 219 y: 208 t: 116
15 x: 0   y: 0 t: 0
16 x: 0   y: 0 t: 0
...
62 x: 0   y: 0 t: 0
63 x: 0   y: 0 t: 0
```

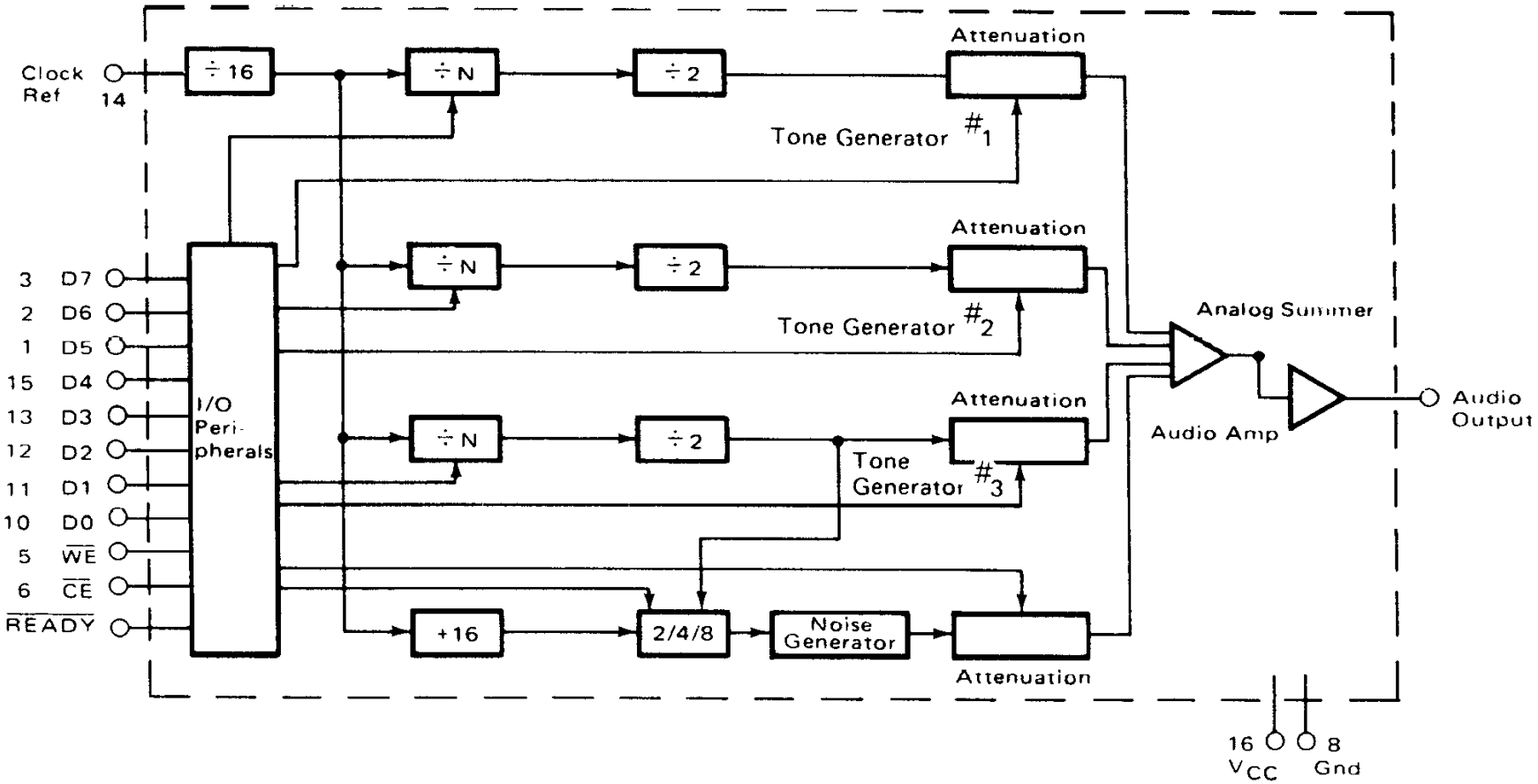
No more sprites marker

Sprites

- “Collision” detection
- Transparency
- Limit of 8 sprites/line



Sound


Clock Ref
14
÷ 16
÷ N
÷ 2
Attenuation
Tone Generator #1
÷ N
÷ 2
Attenuation
Tone Generator #2
÷ N
÷ 2
Attenuation
Tone Generator #3
+16
2/4/8
Noise Generator
Attenuation
Analog Summer
Audio Amp
Audio Output
3 D7
2 D6
1 D5
15 D4
13 D3
12 D2
11 D1
10 D0
5 WE
6 CE
READY
I/O Peri-pherals
16 Vcc
8 Gnd

Sound

- 4 channels
  - 3 square wave
  - 1 noise
- Z80 I/O bus mapped register:

bit	7	6	5	4	3	2	1	0
Volume:	1	r	r	1	d	d	d	d
High freq:	1	r	r	0	f	f	f	f
Low freq:	0	x	h	h	h	h	h	h

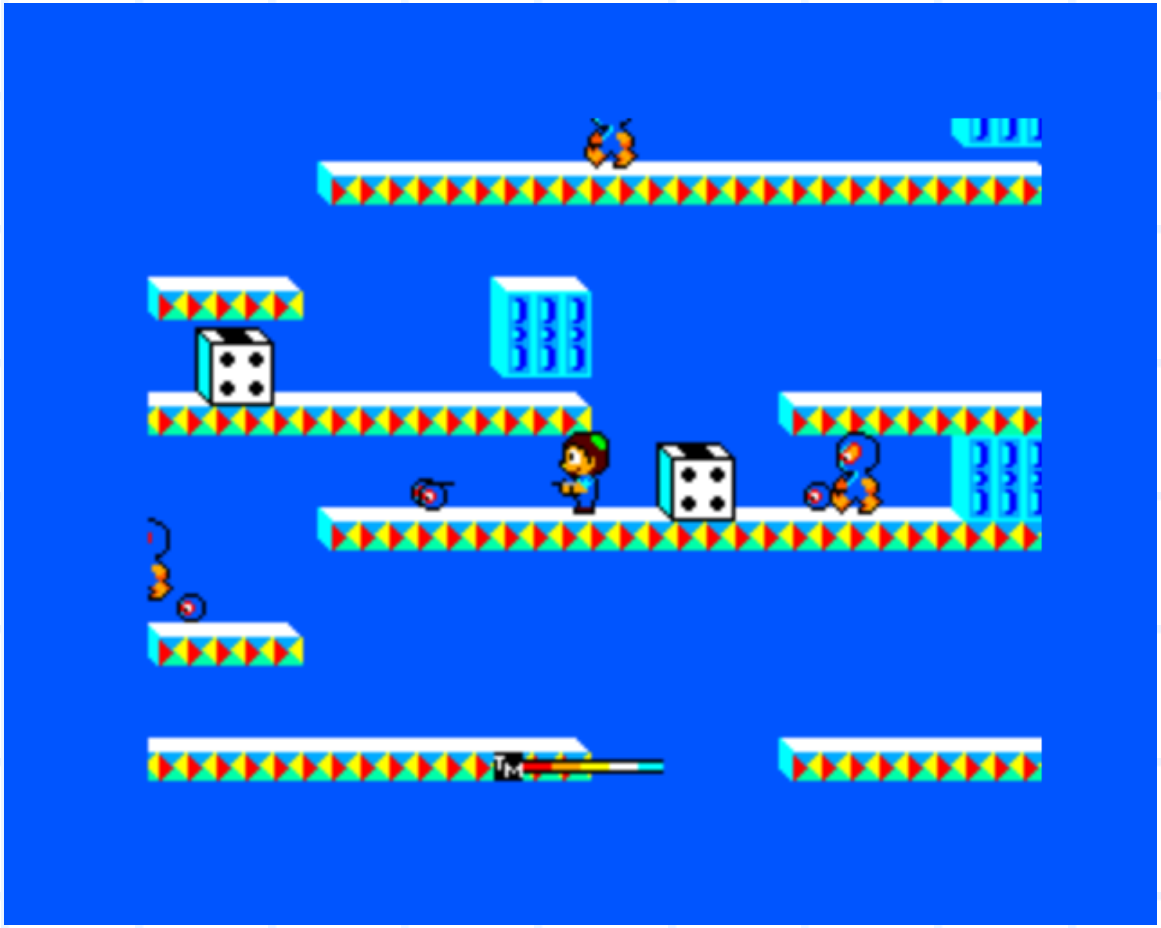
Sound – Tone

```
for (var i = 0; i < length; ++i) {
  counter[chan] -= soundchipFreq / sampleRate;
  if (counter[chan] < 0) {
    counter[chan] += reg[chan];
    out[chan] ^= 1;
  }
  result[i] += out[chan] ? 1 : -1 * vol[chan];
}
```

Sound - Noise

```
var lfsr = 1<<15;
function shiftLfsrWhiteNoise() {
  var bit = (lfsr & 1)
          ^ ((lfsr & (1<<3)) >> 3);
  lfsr = (lfsr >> 1) | (bit << 15);
  return lfsr & 1;
}
```

Putting it all together



Timings – Take 1

```
function main() {
   runZ80(CpuMhz / 50);
   drawVideo();
   generateSound(1/50);
   setTimeout(main, 1000 / 50);
}
```

Timings – Take 2

```
function main() {
  for (var f = 0; f < 313; ++f) {
    runZ80(CpuHz / (50*313));
    rasterizeLine(f);
    generateSound(1/(50*313));
  }
  copyScreenToCanvas();
  setTimeout(main, 1000 / 50);
}
```

Timings – Take 3

```
function main() {
  if (curLine === 0) nextFrame = Date.now() + (1000 / 50);
  run_z80(CpuHz / (50*313));
  rasterizeLine(curLine);
  generateSound(1/(50*313));
  if (curLine++ == 313) {
    curLine = 0; copyScreenToCanvas();
    setTimeout(main, nextFrame - Date.now());
  } else {
    setTimeout(main, 0);
  }
}
```

Optimization

- ***MEASURE FIRST!***
- Table lookups!
- Uint8Array, Uint32Array
- asm.js type constructs
  - `var foo = expr|0;`
- `const bar = …;`

Debugging

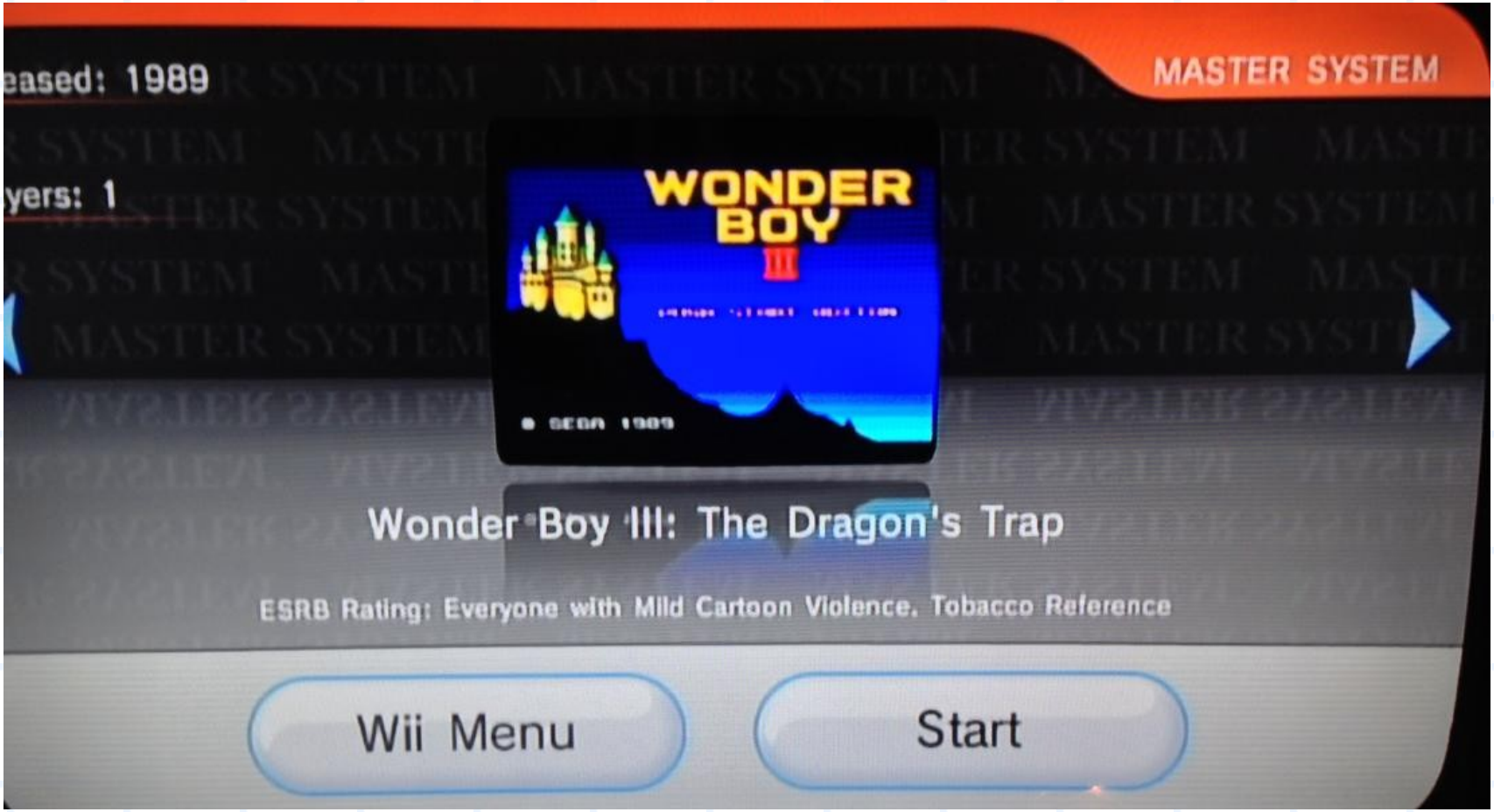
- Built-in debugger
- Example bugs
  - Wonderboy III
  - Fantasy Zone
  - Altered Beast

Future direction

- Test more games
- YM2413 synthesizer
- Save game support
- GameGear
- hqx upscale filter
- More emulators...maybe

Questions?

- http://xania.org/miracle/miracle.html
- http://github.com/mattgodbolt/Miracle
- More info at http://www.smspower.org/
- Some classics available on Wii


eased: 1989
MASTER SYSTEM
yers: 1
WONDER BOY III
SEGA 1989
Wonder Boy III: The Dragon's Trap
ESRB Rating: Everyone with Mild Cartoon Violence, Tobacco Reference
Wii Menu
Start