Game Boy Advance Programming in Forth

Ties Stuij

Overview
What to expect

- The need
- The problem statement
- Resolution?

The need

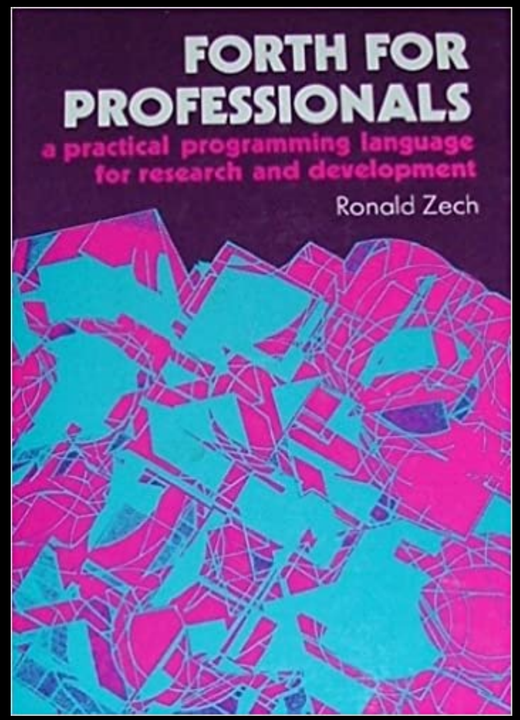
- DS assembler in Lisp
- But what to do with it?

```
104  (defun initialize-and-make-red ()
105    (assemble 'arm9 'arm
106      (emit-asm
107       (blx :main)
108
109       code16
110
111       :main
112       (ldr r0 #x04000000) ; hardware-registers offset and address of reg-disp-ct
113       (mov r1 #x3)                          ; both screens on bits
114       (ldr r2 #x00020000)                   ; framebuffer mode bits
115       (mov r3 #x80)                         ; vram bank a enabled, lcd bits
116       (ldr r4 #x04000304)                   ; reg-power-ctrl
117       (mov r5 r4)                           ; see below
118       (sub r5 #xC4)                         ; 0x04000240 == reg-vram-ctrl-a
119
120       (str r1 (r4 0))
121       (str r2 (r0 0))
122       (str r3 (r5 0))
123
124       (ldr r0 #x06800000)
125       (mov r1 #x31)
126       (ldr r2 #xC000)
127
128       :write-screen-red
129       (strh r1 (r0 0))
130       (add r0 #x2)
131       (sub r2 r2 #x1)
132       (bne :write-screen-red)
133
134       :loop
135       (b :loop))))
```

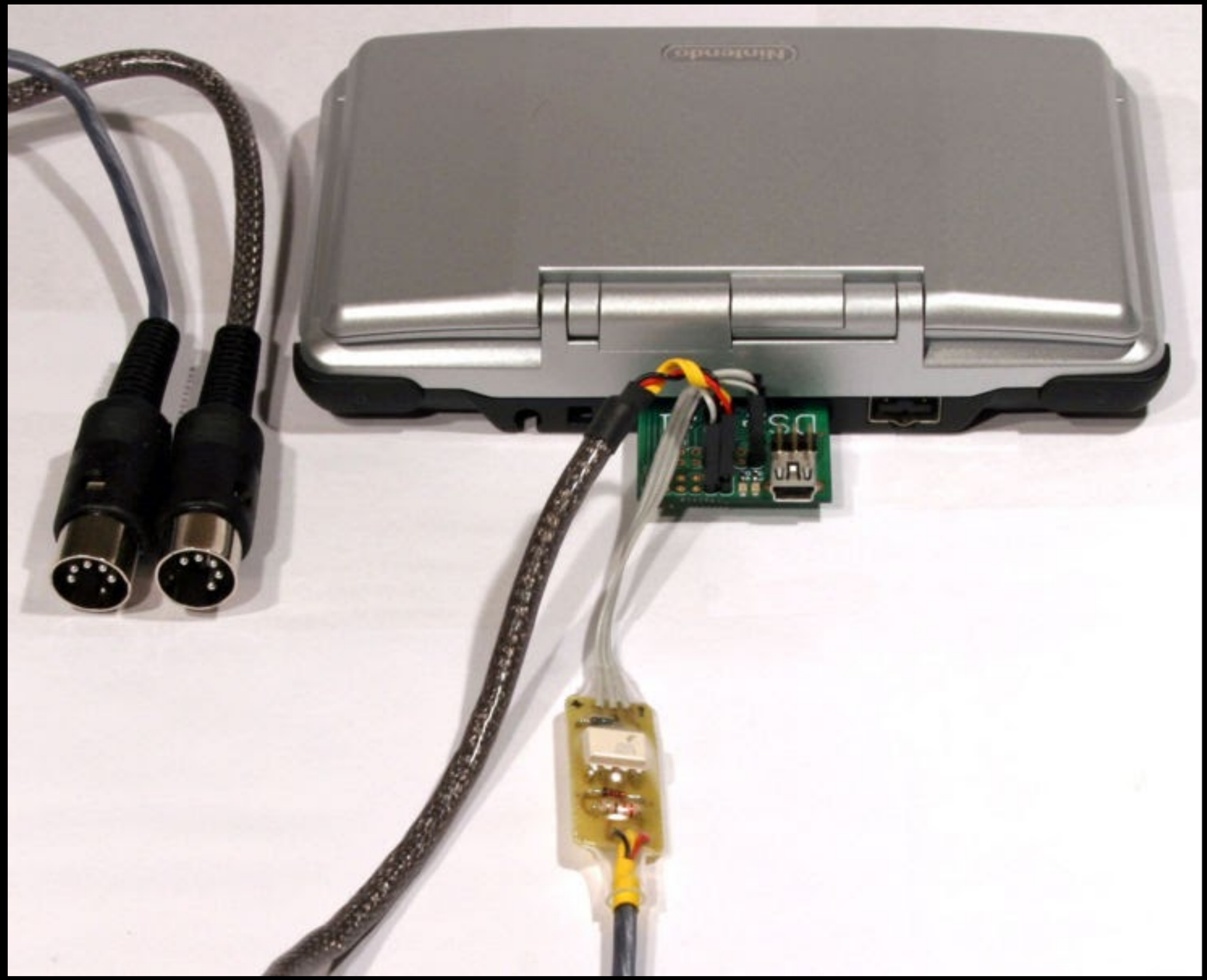
OLE Nepal

One Laptop Per Child

- Open Firmware
- Forth for Professionals
- Jones Forth
- Nintendo DSerial
- Failed



Darkness



New beginnings

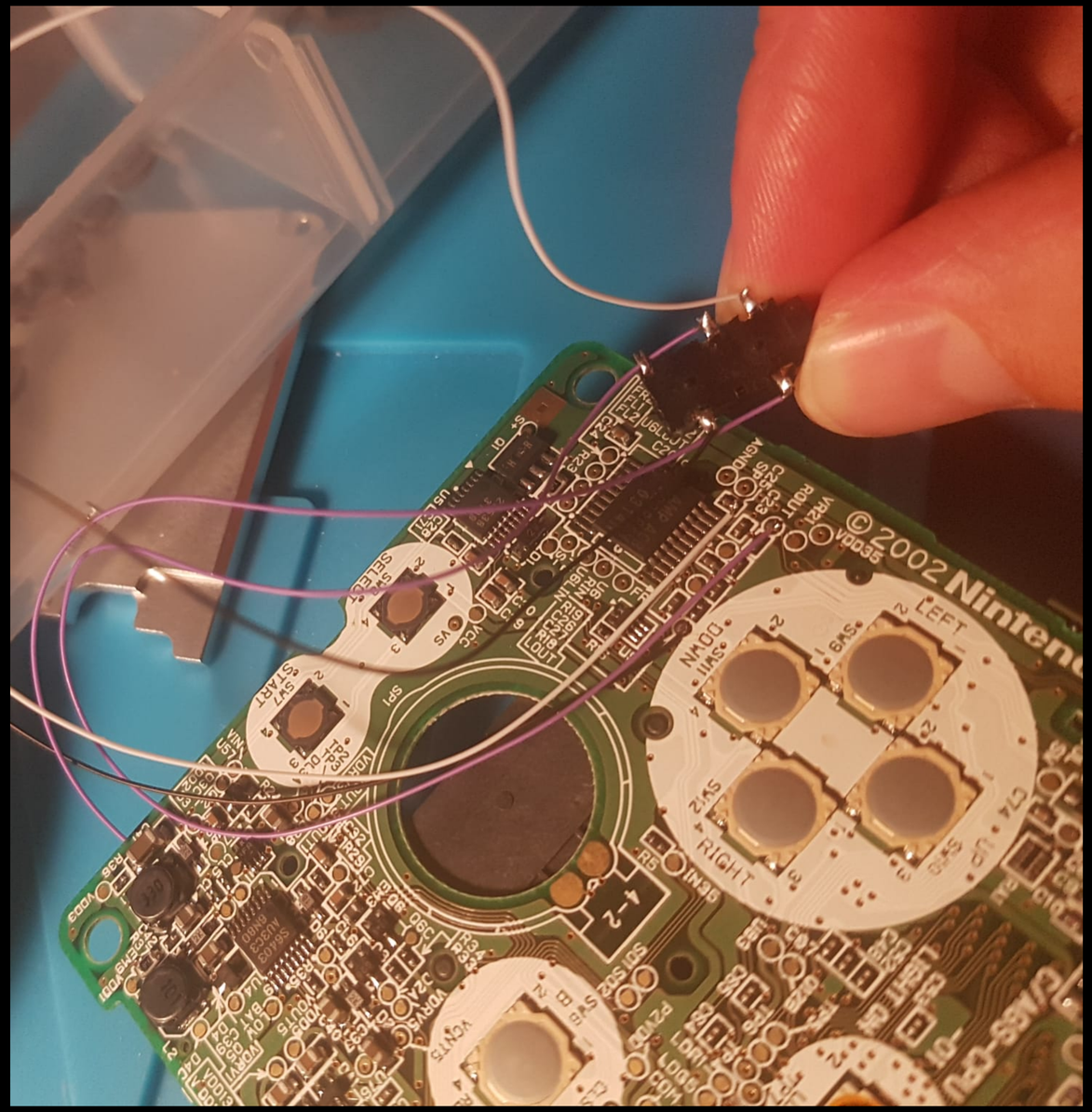
- J1 FGPA Forth
- #gbadev Discord channel
- Idea

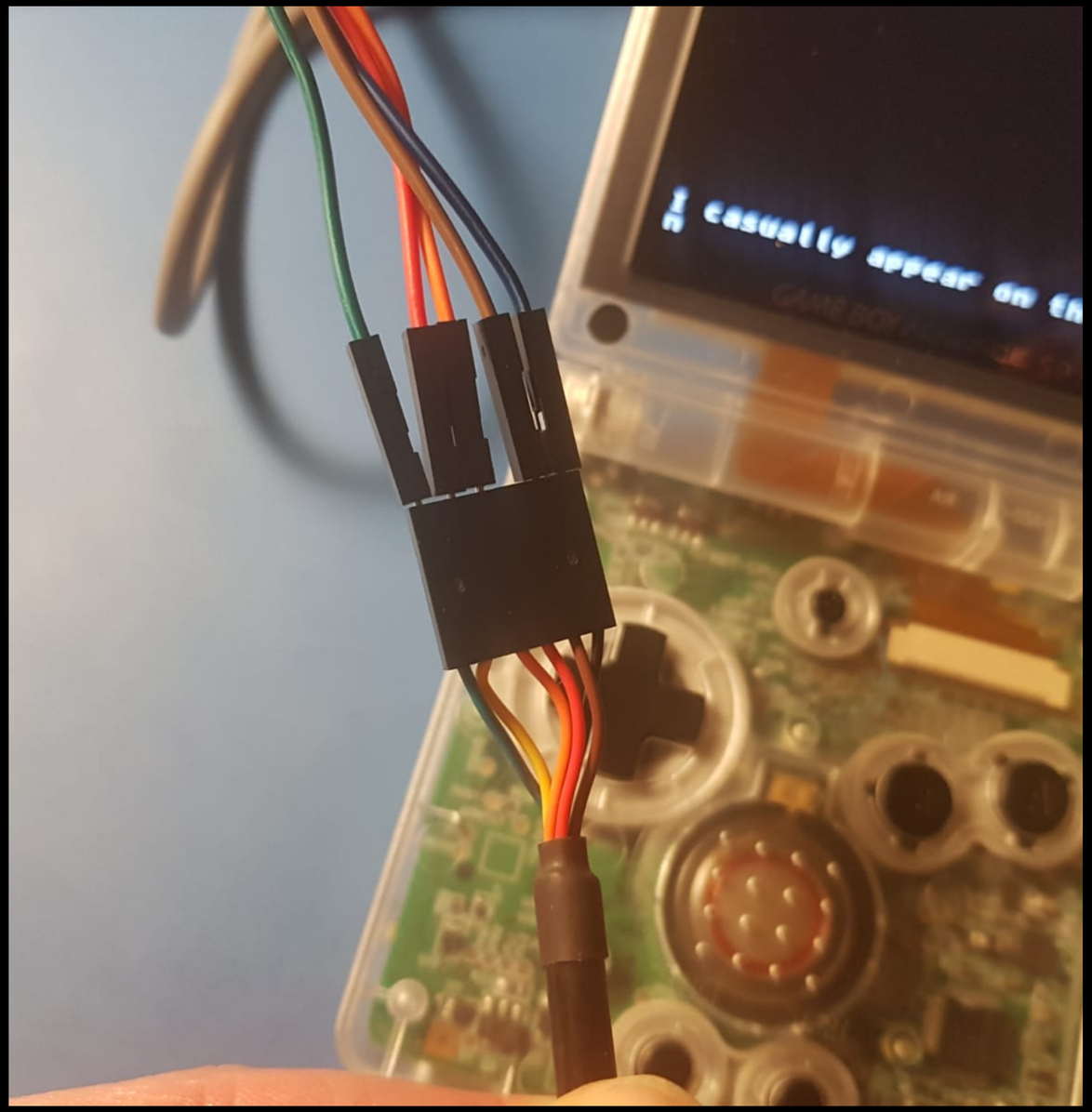
Problem statement

- Need a serial connection
- Need a Forth
- Need a project

GBA internals

- Copetti
- 32-bit Arm 7tdmi at 16Mhz
- vs GB’s 4Mhz Z80 (also on die)
- 3 stage, no cache, no mmu, etc..
- Tiled BG manipulator
- Last of it’s kind (w NDS)
- Registers registers registers
- GBATEK





Need a serial connection

- REPL
- gba-serial-adventures
- Multiple serial modes
- UART!
- Python listener on PC (server)

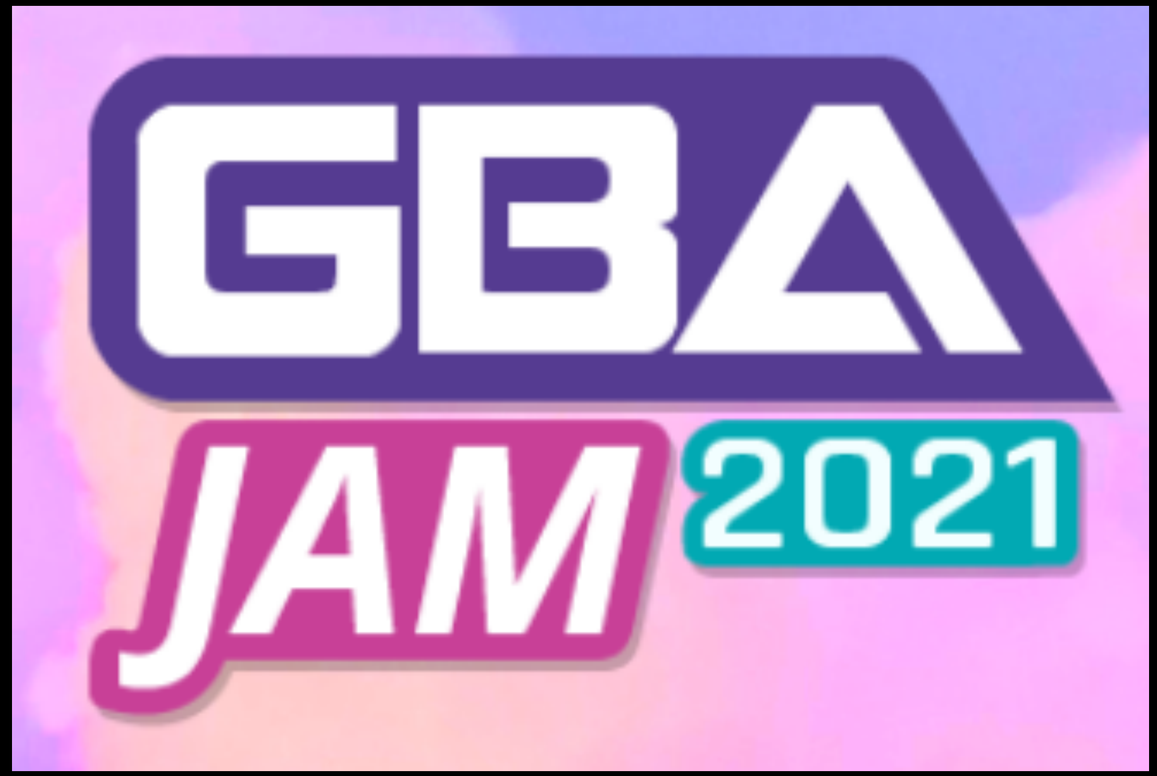
Need a Forth

- Extend own Forth
- What's out there
- Panda in Camel's clothes
- Straight port
- bdos interface to C
- Kernel written in assembly macros

```
949  #C MOVE    addr1 addr2 u --     smart move
950  #             VERSION FOR 1 ADDRESS UNIT = 1 CHAR
951  #  >R 2DUP SWAP DUP R@ +     -- ... dst src src+n
952  #  WITHIN IF  R> CMOVE>        src <= dst < src+n
953  #       ELSE  R> CMOVE  THEN ;          otherwise
954
955  #    head move,4,"move",docolon,within
956  #         .word tor,twodup,swap,dup,rfetch,plus
957  #         .word within,qbranch,move1
958  #         .word rfrom,cmoveup,branch,move2
959  #move1:  .word rfrom,cmove
960  #move2:  .word exit
961
962  #C MOVE    addr1 addr2 u --     assembly move
963      codeh move,4,"move",within
964          ldr     r4, [sp], #4    /* addr2 */
965          ldr     r5, [sp], #4    /* addr1 */
966      cmp r1, #0
967      ble move2
968  move1:
969      ldr r6, [r5], #4
970          str     r6, [r4], #4
971      subs r1, r1, #4
972      bgt move1
973  move2:
974          ldr     r1, [sp], #4    /* pop new TOS */
975          next
```

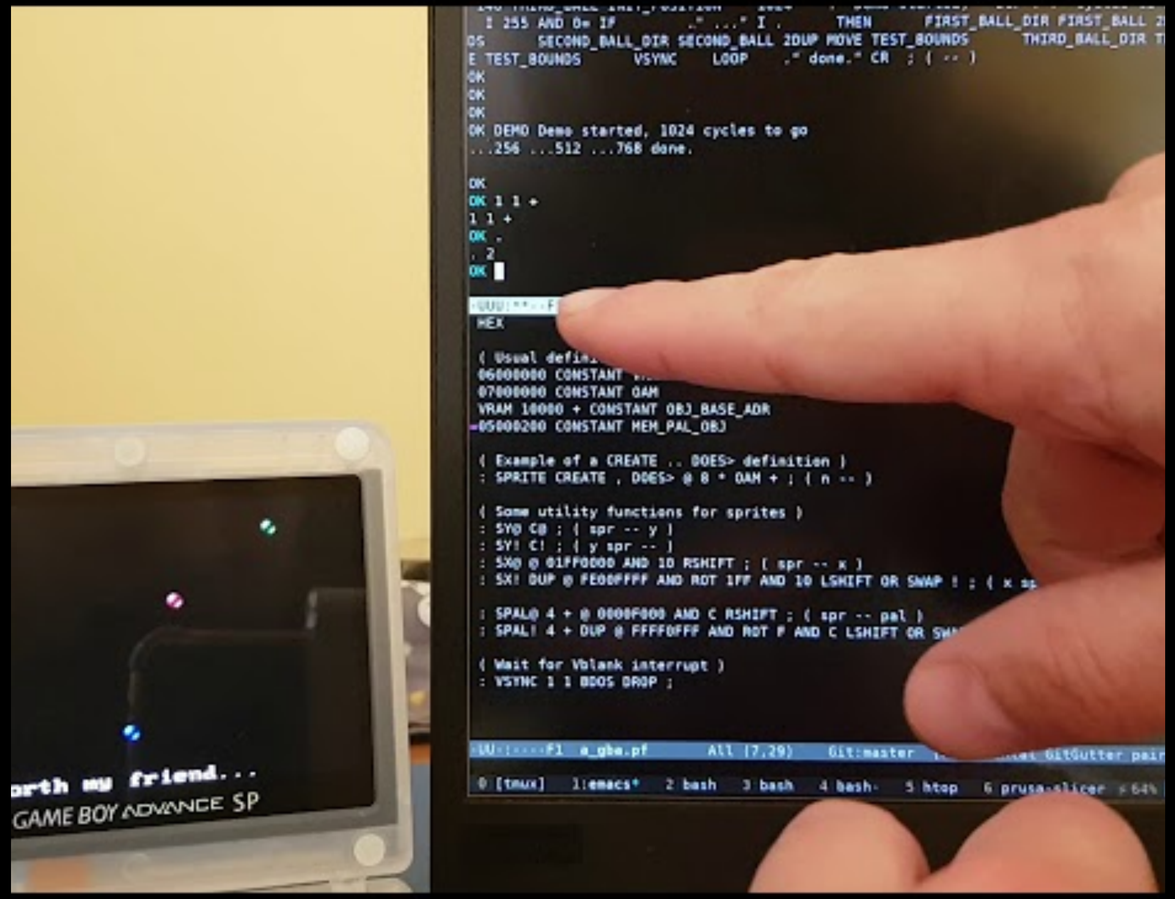
Need a project

- GBA Game Jam
- Games need all your cycles
- 60fps refresh rate
- Update sound (software mixing), graphics, game logic
- Normally in C, but sound in asm
- 3 months



Putting it all together

- Load program on ROM cart
- Emacs Forth mode
- REPL
- Compile text or snippets on GBA



Trouble in paradise

- No GBA specific code (yay!)
- Compiling on GBA SLOOOWWW
- We’re on a deadline. Let’s hack together something ugly.
- It kinda works, but oh you harsh mistress immediate mode

```
294  def colon_compile(context):
295      words = []
296      tokens = context.tokens
297      definition = [tokens.pop(0), tokens.pop(0)]
298      name = definition[1].tok
299      branch_stack = []
300      if_stack = []
301
302      next = peek(tokens).tok
303      while next != ";":
304          try:
305              words.append(parse_number(context))
306              next = peek(tokens).tok
307              continue
308          except ValueError:
309              pass
310
311          if next == "(":
312              parse_comment(context)
313          elif next == "begin":
314              label = Label(name, tokens.pop(0))
315              words.append(label)
316              branch_stack.append(label)
317          elif next == "until":
```

Done!

- It kinda works
- Really fun
- Interesting problems to solve
- Rath
- https://github.com/stuij/rath