Let’s Play(AI)
Super Metroid

swarley
kronicd
omnifocal

Agenda

- Who are we?
- Intro to console emulation and the Super Nintendo
- Intro to ML/AI/New coolness
- Mar/IO
- AI driven Super Metroid
- Conclusion and questions (maybe?)

whoamwe?

- swarley (@swarley777)
  - academic turned hacker working at Asterisk
- omnifocal (@omnifocal)
  - academic turned hacker working at Asterisk
- kronicd (@kronicd)
  - ~~hacker turned~~ academic ~~turned hacker working at Asterisk~~

real quick...

- The videos playing throughout this presentation are of our AI smashing various bosses in Super Metroid

- ***Also: none of us are experts in machine learning***
  - we can't even read tbh

Console emulation + SNES

SNES

- Very, ***VERY*** good console Nintendo console released November 1990
  - 49mil sold (thanks Wikipedia!)
- 16 bit colours and sounds
  - WHOA!
- Cartridge-based games
- kronicd and I traded our's to cashies :(
  - Realtalk: that stings more than not buying BTC

SNES Emulation

- Dump ROM from game cartridge; play on a general-purpose computer!
- Support some degree of save states and reloads, maybe rewind, game genie cheats, etc.
- Multiple emulators on the market
- Some chip-perfect copies of the SNES console, others are less faithful
    - BSnes
    - ZSNES
    - SNES9x
    - etc.
- BizHawk has Lua bindings and a TAS console (important later)

ML done basic

- Computer gets input; computer guesses output
- Guessing is achieved through different ML algorithms
  - These algorithms vary wildly in operation and training schedule
- Typically supervised, unsupervised or reinforced

ML problem domains

- Supervised
  - Classification: sort input data into pre-defined classes. Trained using pre-classified data (discrete data)
  - Regression: predict the weather based on previous weatherz (continuous data)
- Unsupervised
  - Clustering: sort unknown data into clusters of similar dataz
  - Feature extraction: reduce input data down to key descriptors/ features
    - Typically fed back into another algorithm
- Reinforcement learning
  - Robots: robots

Neural networks

- Buzzword for most machine learning algorithms
- Very commonly used
  - First example on Google
- Simulates "human brain" by creating a network of "neurons" between input(s) and output(s)
- Typically used for pattern recognition tasks
  - Actual use cases are ***limitless***(tm)
  - Cylance lol

Evolutionary algorithms

- Genetic algorithms, etc.
- Goal is to generate the optimal genome/species
- Simulate the properties of organic evolution
  - population of items breed (CROSSOVER)
  - survival of the fittest / natural selection
  - genetic mutation (MUTATE)
  - ...lot's of very weird (and disturbing) family trees
- Used for search optimisation problems
  - Can be also used to determine optimised variable parameters

Fitness functions

- Heart of any evolutionary programming algorithm
- Determines how "fit" (good) the resulting genome is
- Assessed based on some property/ies, e.g.:
  - "Mario beat the level!" == More fit/more good
  - "Time of completion was lower" == More fit/more good
  - "Mario died :(" == Less fit/Mario sucks; bin this one

Mar/IO

- SethBling used unsupervised machine learning to train AI to play Super Mario World.
  - Incidentally - the dude currently holds a number of SMW speedrun world records
- Genetic neural network employed
  - Genetic algorithm trains neural network which performs the actual "playing" of the game
  - Button presses are decided in the Neural Network based upon the input data provided (i.e. the objects on screen)
  - Genetic algorithm assesses the "fitness" of the current playthrough and breeds accordingly
- Eventually Mario became pretty good at the game

Mario is easy; Can it play Super Metroid?

- Super Metroid (1994) = Best game ~~on SNES~~
- Objectives vary SUBSTANTIALLY from Super Mario World
- Not just "get to end of level/run right!"
- Traverse sprawling maps, backtrack
- Fight bosses
- Conserve ammunition where required, use liberally elsewhere
- Move up/down/left/right through individual rooms
- Kill enemies, or don't. It depends on the situation!


AUTO
ENERGY 99
060
15
20
frozen enemy
energy and ammo
metroid
beam shot

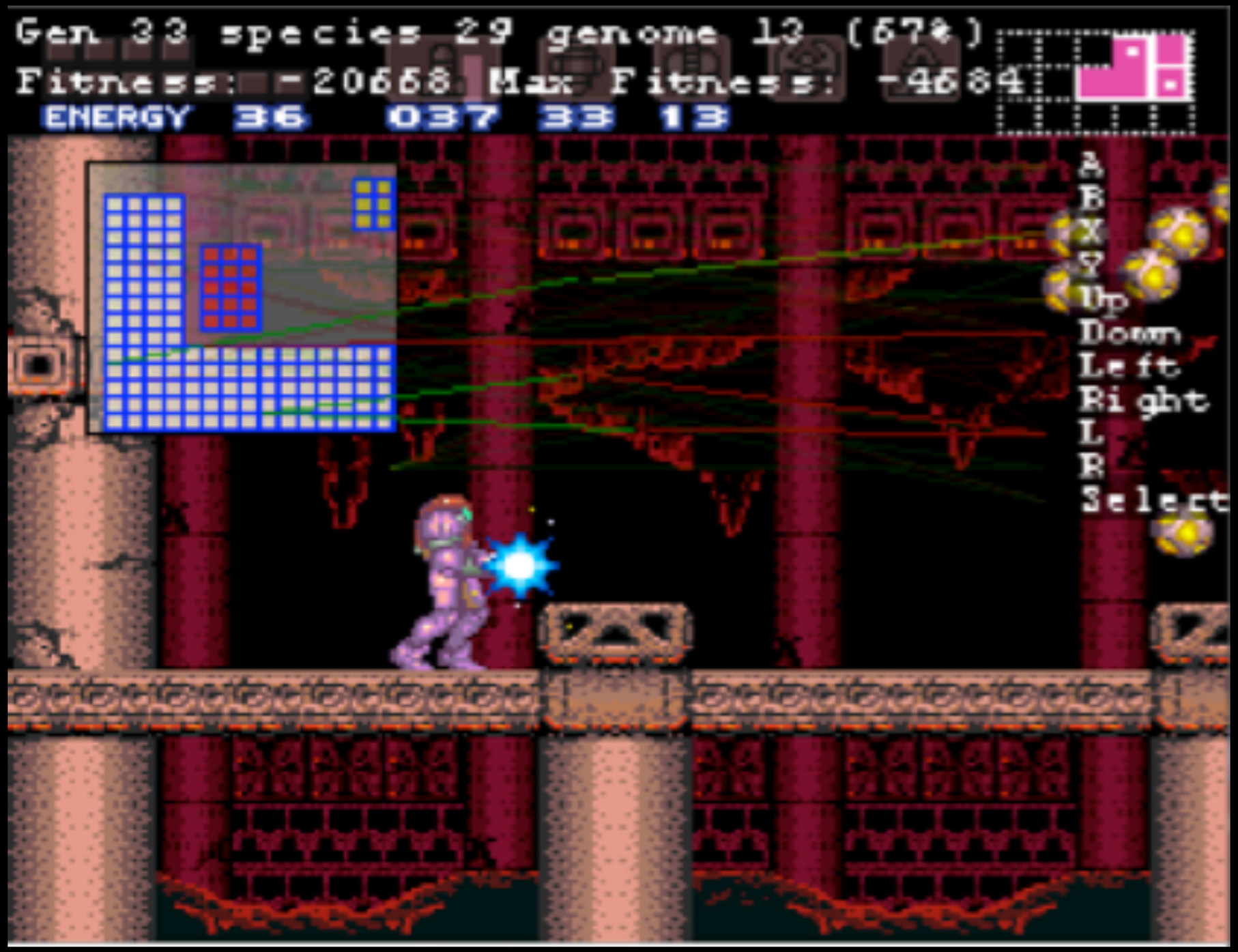
how we fixed metroid

- Create species/genomes
- Run ANNs (more to come)
- Cull the worst ones
- Fittest species breed
- Rinse and repeat until memory leaks become extreme

how we fixed metroid

- Every frame
  - Build inputs (feATURE EXTRACTION BUT NOT)
- Evaluate ANN
  - For inputs produce outputs
- pushg buttans
- Evaluate fitness
- Extend timer if appropriate

shaowmap


Gen 33 species 29 genome 13 (67%)
Fitness: -20668 Max Fitness: -4684
ENERGY 36 037 33 13
A
B
X
Y
Up
Down
Left
Right
L
R
Select

ack jack

```lua
function hasMissiles()  -- Return 1 if metroid has missilez, otherwise 0
    local missiles = mainmemory.read_u16_le(0x0009C6)
    if missiles > 0 then
        missiles = 1
    end
    return missiles
end
```

kronicd <kronicd@kronicd.net> 8/20/17

to Kejardon

Hi,

I'm attempting to port MarI/O to work with Super Metroid. I've got most of the inputs I need mapped out, however I'm missing one important detail and considering your significant work on reversing the game, I thought you might have some ideas.

Your documentation shows:

```
7E:1A4B - 7E:1A6E    X position of projectile, in pixels
7E:1A6F - 7E:1A92    X subpixel position? (guessing)
7E:1A93 - 7E:1AB6    Y position of projectile, in pixels
7E:1AB7 - 7E:1ADA    Y subpixel position? (guessing)
```

However it seems 32 items are stored in this region, including both enemy projectiles, power ups and non-interactive window dressing items (dripping etc). Additionally the locations remain stored here after the item has vanished.

Do you have any idea where the details on *what* each item found in this range is?

I've included a couple of screen shots below showing the coordinates gathered from these addresses in white. As you can see there are a few different types of thing are shown here.

Any help would be great :)

Cheers

Kevin Swanson <kejardon@gmail.com>

8/20/17

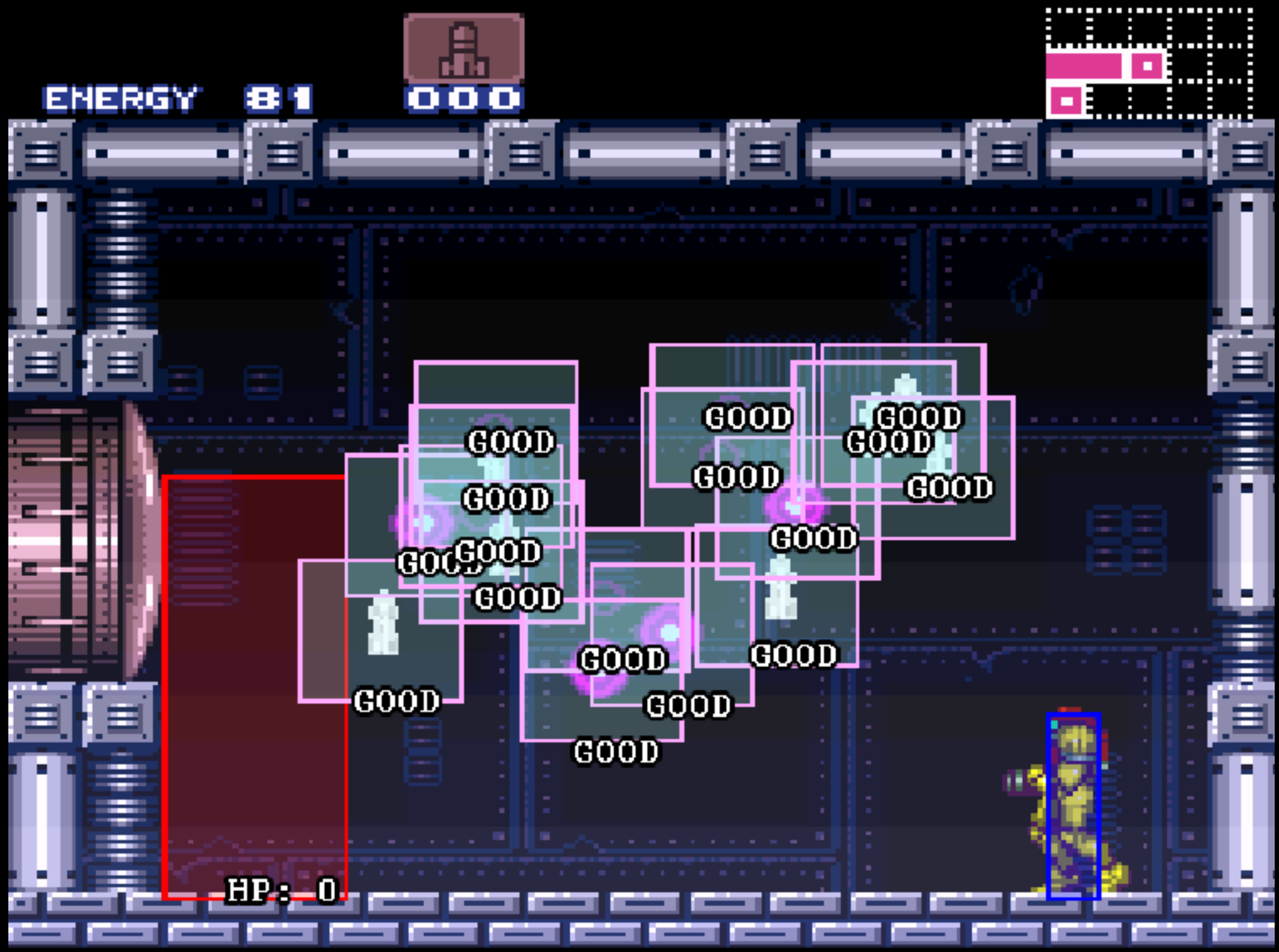
to me

I'm not familiar with Marl/O so I don't have any advice specifically for that.
'Room projectiles' is the general term I use for a lot of things like that. It covers a LOT of stuff, yes. The RAM used for projectiles is everything from 7E:1995 to 7E:1C1E. With tables like this in Super Metroid, it's the header (7E:1997 - 7E:19BA) that determines if the object actually exists or has been deleted. This same header also tells you what type of object it is. In RandomRoutines.txt, there's a section of code focused on these:

86:8027 spawns enemy specific projectiles (0F96/0F98 copied to projectile).
86:8097 spawns enemy/room objects (index in Y) (0 used for pallete and sprite index). Used by some enemies, also gate PLMs.
86:8104 is the main enemy/room projectile engine. Called every game frame.

There's also a file, ERProjectiles.txt, that has basic info about how they work and a few examples. I'm attaching that in case you don't have it for some reason. I don't have a complete list of all the things that use it though, sorry.

Debug screenz



ENERGY 69

010

BAD

BAD

HP: 20

BAD

HP: 20

ENERGY 25

000

GOOD

BAD

BAD

BAD

BAD

BAD

BAD

BAD

HP: 160

Fitness function

- Metroid is rewarded for:
  - Killing or hurting enemies
  - Collecting pickups (health, ammo, etc.)
  - Living
- He is also punished for:
  - Standing still for extended periods of time (he really likes to do that)
  - Taking damage / running out of health

Fitness function Cont.

- This works well enough for some "boss" scenarios
- Unfortunately other scenarios are not so straightforward
- future_work.jpg

Teh enemies

Enemies that
metroid fight gud

‘Charizard’ - The Robot

- Vulnerable mid section
- First minor boss
- Weak

The Worm

- Fast moving
- Head always vulnerable
- Moves in and out of field

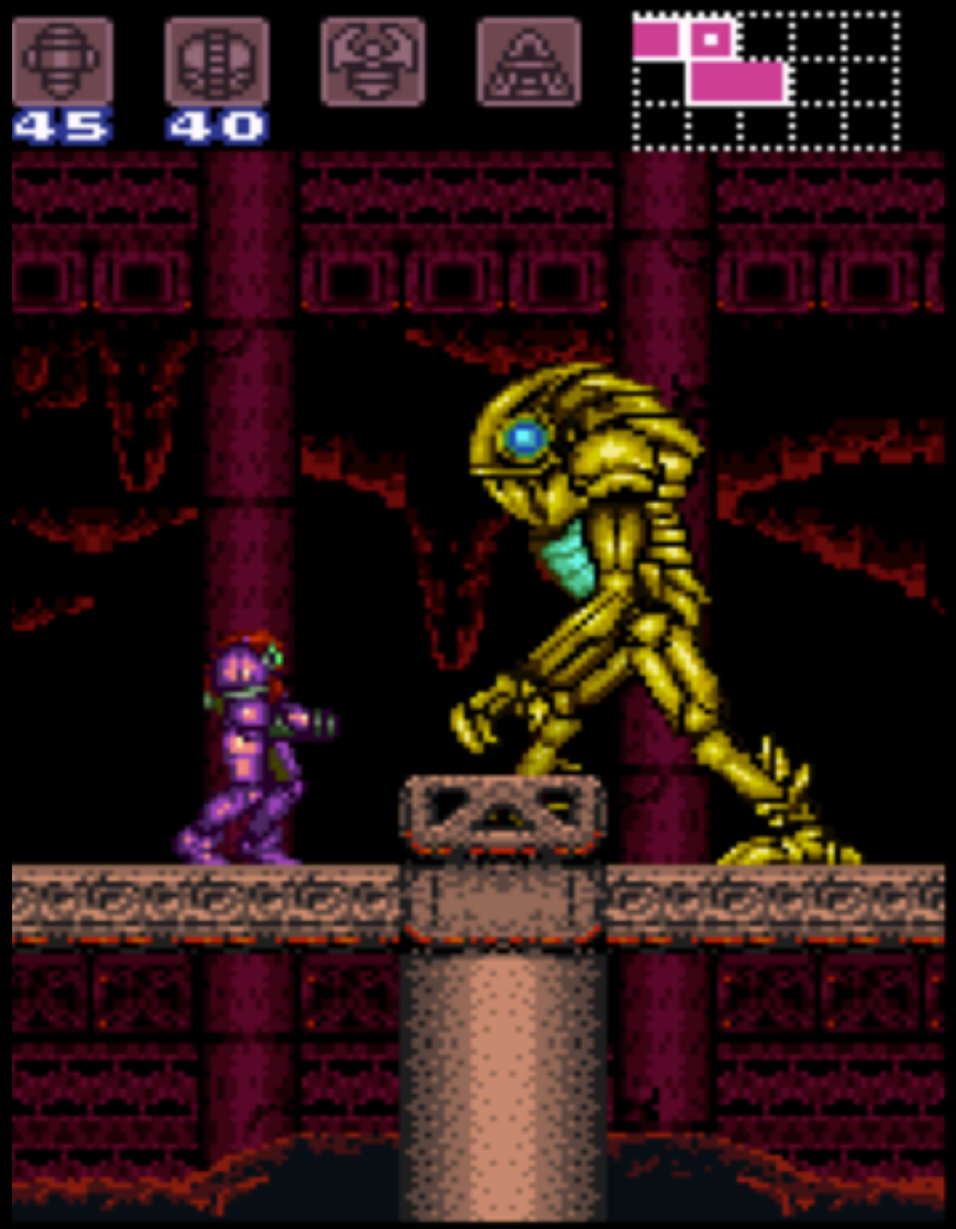
Enemies that metroid are ok at fight

"Ripley" - The Robot Bird

- Advanced strategy
- Extremely strong
- Invulnerable tail

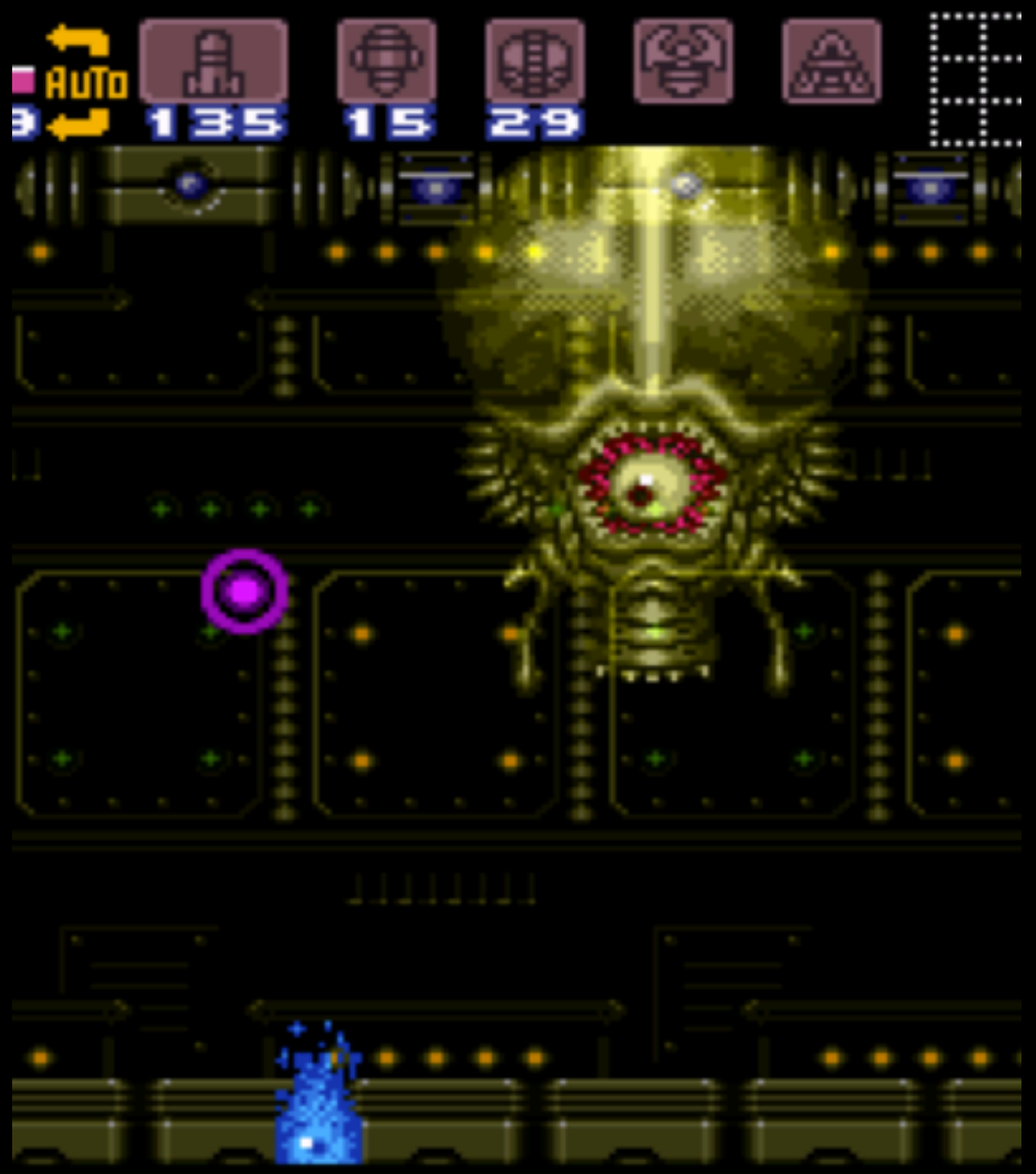
Golden charizard

- Same as the other charizard
- Bit stronger
- Don't use super missles
  - (he gets super pissy)



Enemies that metroid fight not so good

"Robert"



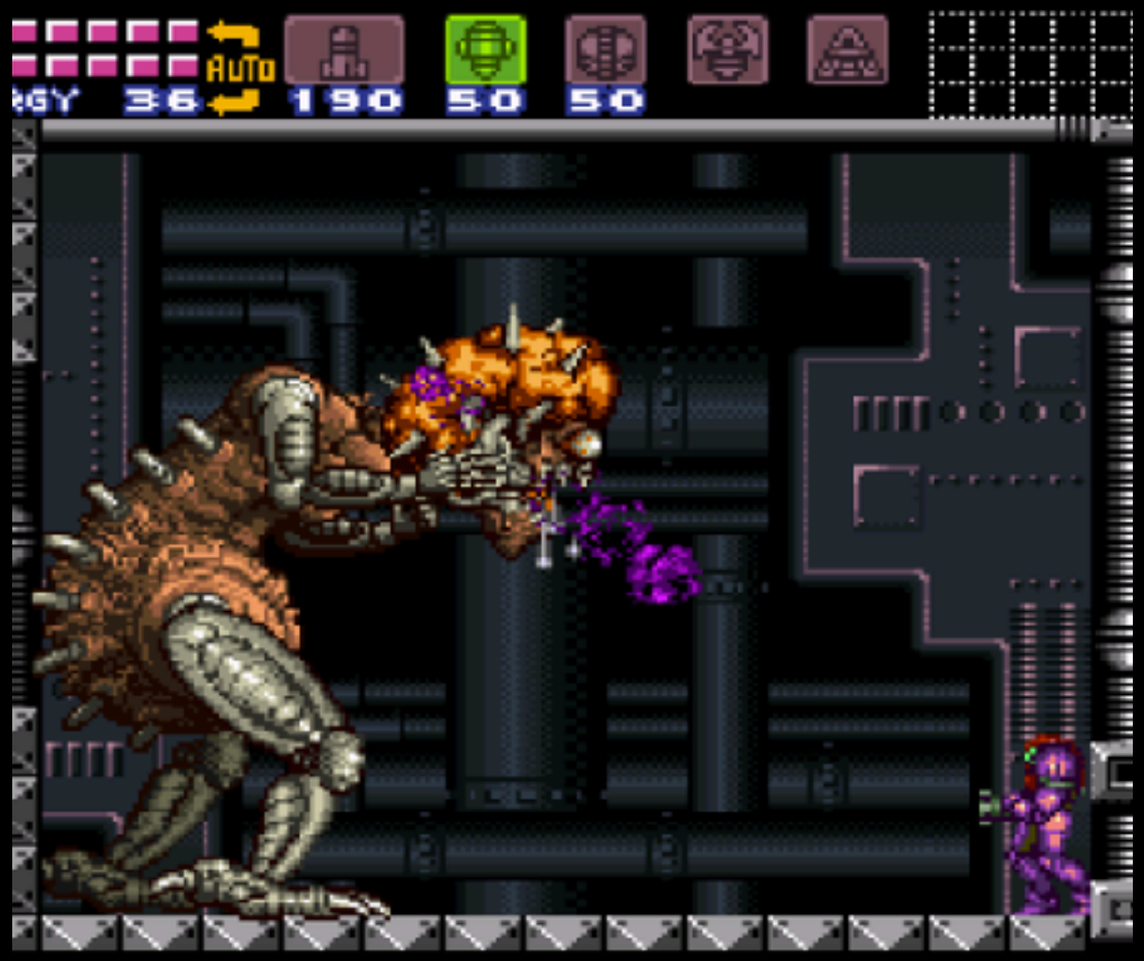
Clyde

Spike ball

Croco

"Bulbasaur"

Day-man



Challenges

Lua

- Seriously. Lua sucks.
  - Who the hell 1 indexes?!
- The code for Mar/IO was not amazing
  - We made it arguably... worse

```lua
function writeFile(filename)
    local file = io.open(filename, "w")
    file:write(serpent.dump(pool))
    -- file:write(pool.generation .. "\n")
    -- file:write(pool.maxFitness .. "\n")
    -- file:write(#pool.species .. "\n")
    --     for n,species in pairs(pool.species) do
    --     file:write(species.topFitness .. "\n")
    --     file:write(species.staleness .. "\n")
    --     file:write(#species.genomes .. "\n")
```

```lua
--- main garbage to run

-- SETUP
-- build pool, this was randomly between functions ffs
-- MAIN LOOP
if pool == nil then
    inputs = getInputs(True)
    initializePool()
end
```

```lua
for y=0,14 do
    for x=0,16 do
        -- for when garbage data causes extend blocks that make infinate loops
        TileX, TileY = (x )*16 - bit.band(cameraX, 0x000F), (y+2)*16 - bit.band(cameraY, 0x000F)
        -- this if for pixel-aligning the grid, because the screen doesn't just scroll per block!
        a = bit.rshift(bit.band(cameraX+(x)*16, 0xFFFF), 4) + bit.band(bit.rshift(bit.band(cameraY+(y)*16, 0xFFF), 4)*(width), 0xFFFF)
        BTS = 0x16402 + a
        -- BTS of block
        Clip = 0x10002 + a*2
        local wall_type = bit.rshift(mainmemory.read_u16_le(Clip), 12)
        if wall_type ~= 0x0 then
            wallGrid[#wallGrid+1] = {["x"] = x, ["y"]=y+2} -- lol? idk why but this ACTUALLY aligns the fucking thing correctly on the y-axis :/
            -- x axis is still fucked though, nfi how to fix that
            tileTypes[(x) + ((y+2) * 16)] = wall_type
        end
    end
```

```lua
function evaluateNetwork(network, inputs)
    --table.insert(inputs, 1) -- this was commented out in our stuff, but not the original, not sure why -- i see why, fuck knows why it was there
    -- this was commented out because it seems to only exist to fix the inherent off by one errors everywhere
```

```lua
-- idk?
-- fucking math eyyy
function sigmoid(x)
    return 2/(1+math.exp(-4.9*x))-1
end
```

```lua
-- idk what this bit does
```

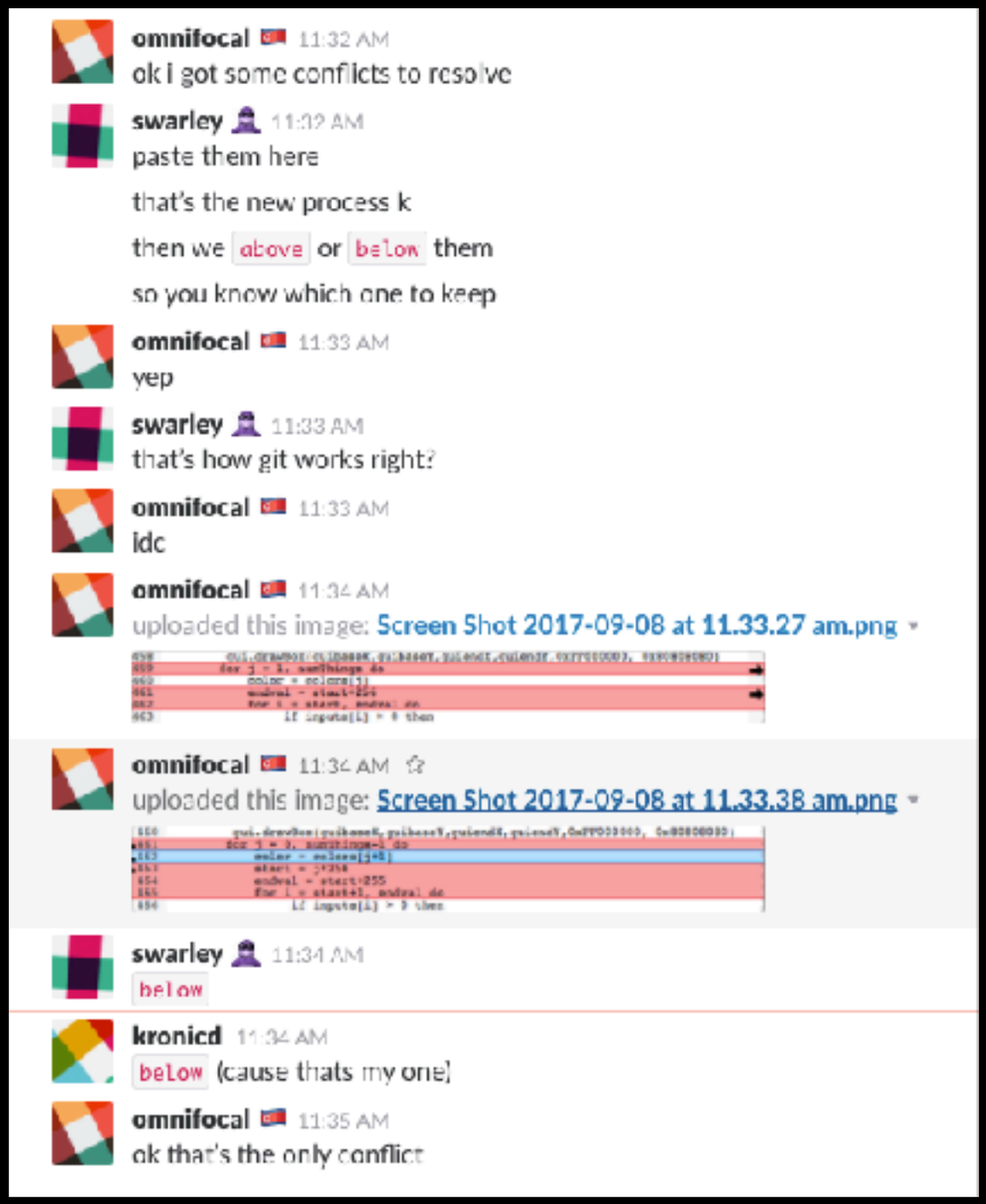
```lua
-- add special shit here -- todo display this shit
```

```lua
--------------------------------
--     END CLOCK FUCK     --
--------------------------------
```

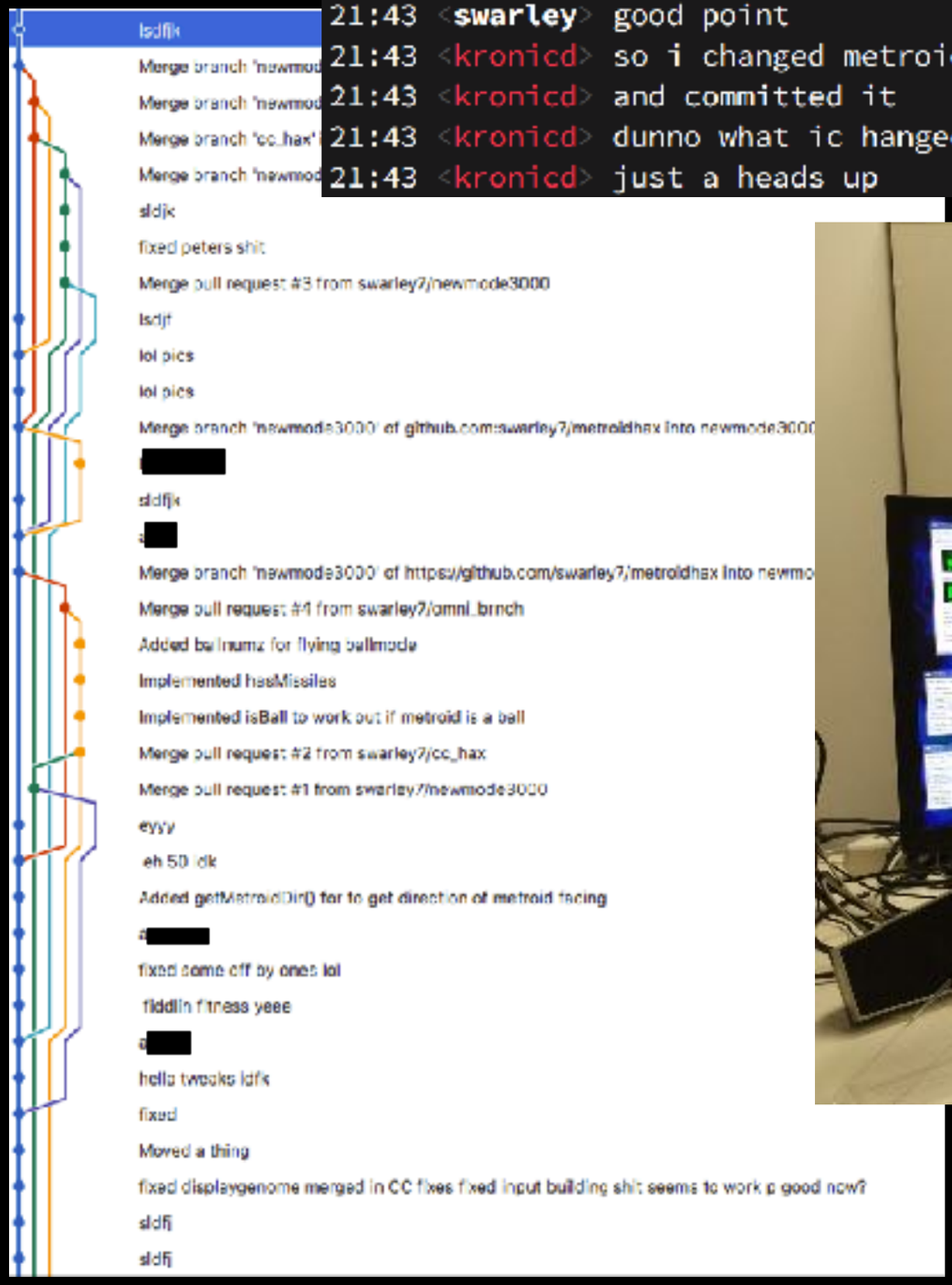
```lua
--if pool.currentFrame%1 == 0 then -- seems to run every 5 frames, that's probably ok?
    evaluateCurrent() -- run every frame maybe we can ease off later idk. metroid is p fast
--end
```

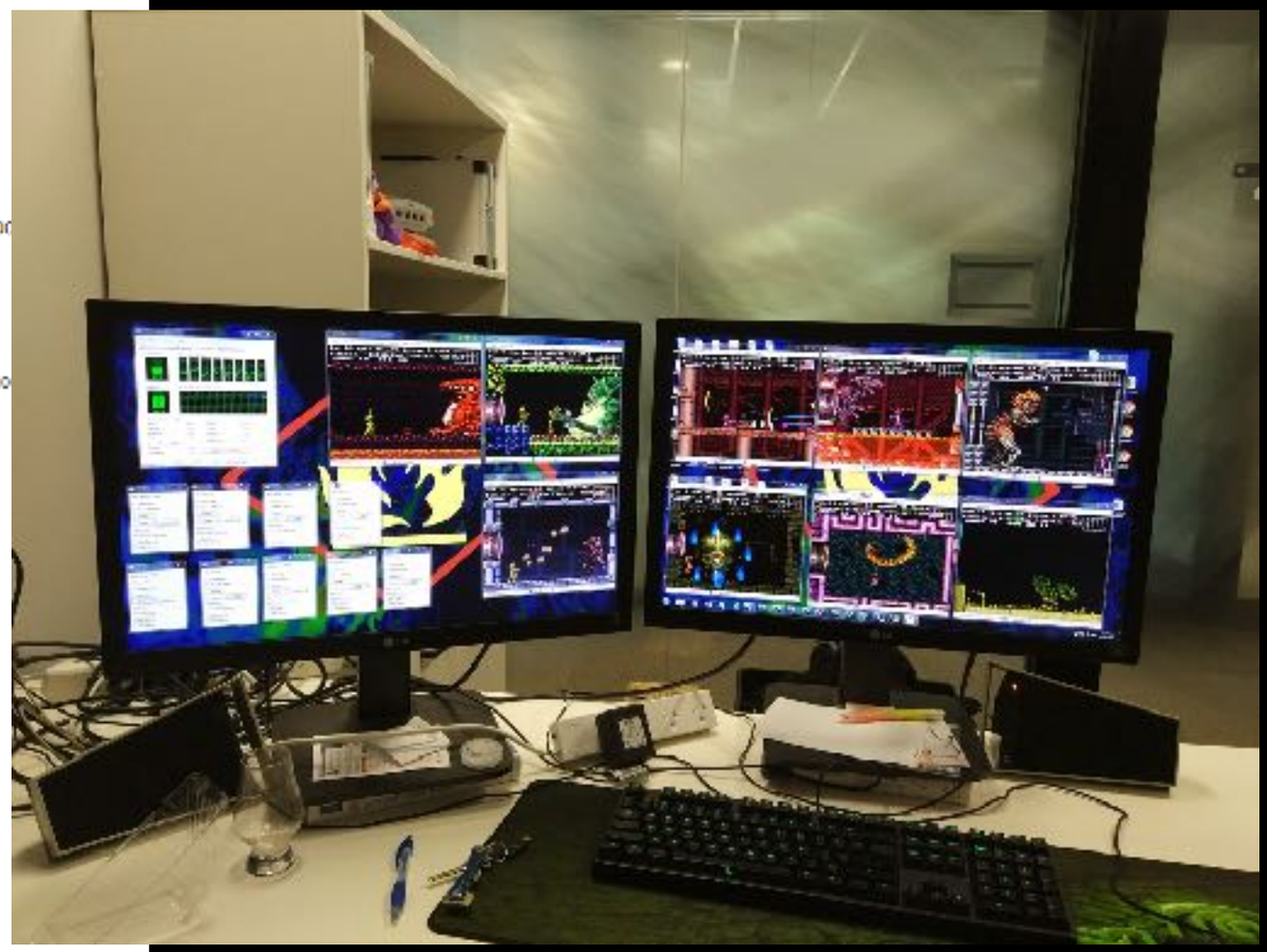
git

- Git is amazing
- unless you use it intentionally wrong for fun
- ...and later decide you'd prefer to do it right



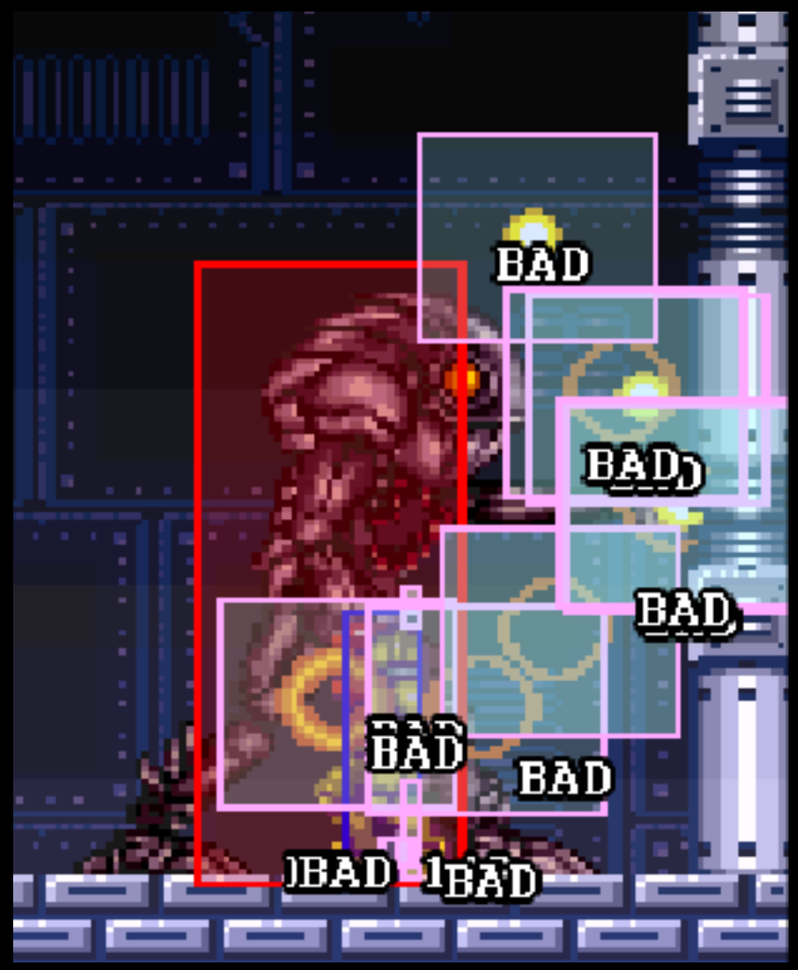
```
21:42 <swarley> and doesn't work
21:42 <swarley> but yeah
21:42 <kronicd> fuck
21:42 <kronicd> im basing all my lua knowledge off reading syntax from other metroid scripts and mari/o
21:42 <kronicd> why the fuck would i need to know anything else
21:43 <swarley> ye
21:43 <swarley> good point
21:43 <kronicd> so i changed metroidlol.lua somehow
21:43 <kronicd> and committed it
21:43 <kronicd> dunno what ic hanged
21:43 <kronicd> just a heads up
```





BizHawk

- BizHawk is a widely used emulator for TAS as it supports Lua scripting and has built in tool-assist features
- It's also garbage
  - Only runs on Windows
  - Very flaky
  - Memory leaks bad



kronicd's desktop at work

Figure 1: an appropriate ~~mis~~appropriation of university resources

Future suggestions

- Completely headless emulator
  - Saves GPU/CPU/graphics
- Cross-platform written in a sane language
  - Python bindings?
  - Better code(ers)
- Try different learning systems
  - GNN is basically worse than brute force imo

QUESTUIONS?