

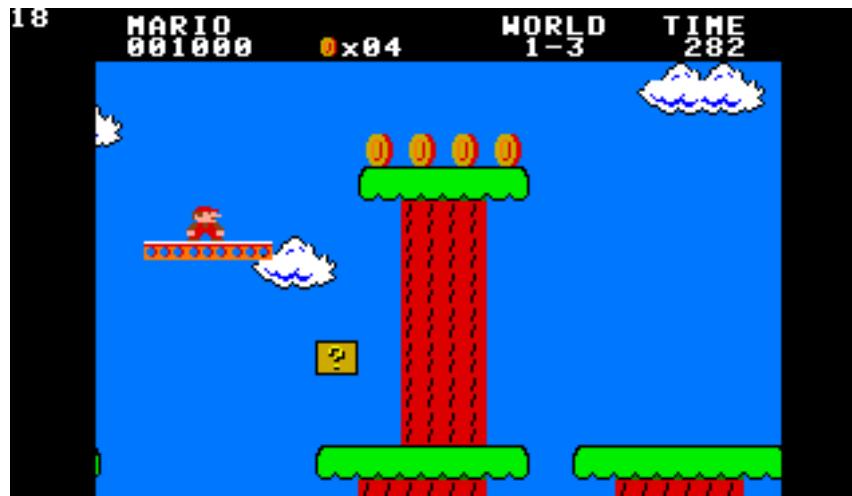


# Generic TILE ENGINE



# What is GTE?

- **Generic Tile Engine**
  - **Generic:** Useful for many types of games
  - **Tile:** Supports tile-based graphics inspired by 8/16-bit consoles
  - **Engine:** Vroom! Vroom!
- Project Page: <https://github.com/lscharen/iigs-game-engine>
- Documentation: <https://lscharen.github.io/iigs-game-engine/toolboxref.html>



# Goals

- **Flexible**
  - Minimal restrictions
- **Features**
  - Leverage the Ilgs' quirks
  - A few clever techniques
- **Fast Enough**
  - Can't guarantee 30/60 fps
  - Easy to implement something fun



# Preliminaries

## • Code

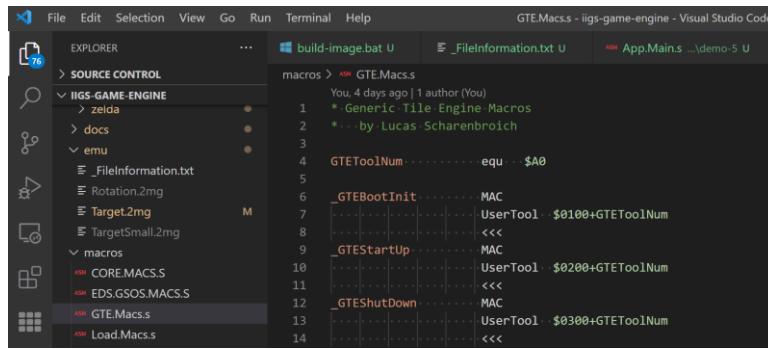
- All 65816 Assembly
- VS Code + Merlin32 + Emulator
- Toolbox

## • Content

- Tiled for map editing
- Paint.Net for tilesets
- Node.js scripts to build assets

## • Debugging

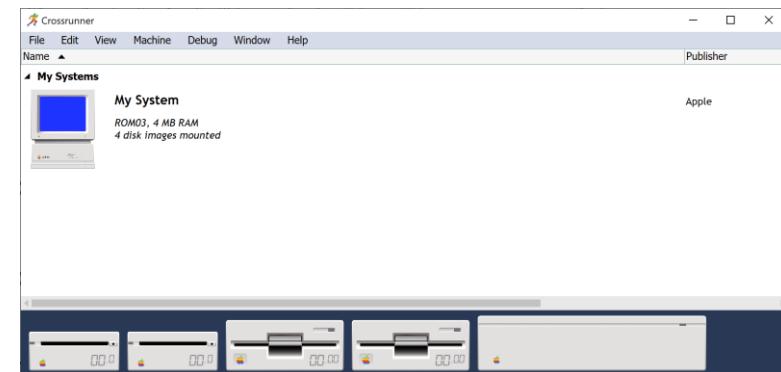
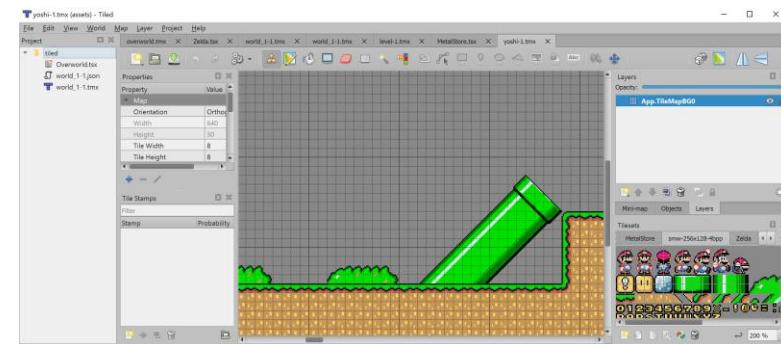
- Crossrunner
- brk



```

File Edit Selection View Go Run Terminal Help
build-image.bat U _FileInformation.txt U App.Main.s ...\\demo-5 U
GTE.Macs - iigs-game-engine - Visual Studio Code
EXPLORER SOURCE CONTROL
IIGS-GAME-ENGINE
zelda
docs
emu
FileInformation.txt
Rotation.zmg
Target.2mg
TargetSmall.2mg
macros
CORE.MACS.S
EDS.GSOS.MACS.S
GTE.Macs.s
Load.Macs.s
macros > GTE.Macs.s
You, 4 days ago | author (You)
1 * Generic Tile Engine Macros
2 * ... by Lucas Scharenbroich
3
4 GTEToolNum ..... equ ... $A0
5
6 _GTEBootInit ..... MAC
7 UserTool $0100+GTEToolNum
8 <<<
9 _GETStartUp ..... MAC
10 UserTool $0200+GTEToolNum
11 <<<
12 _GETShutDown ..... MAC
13 UserTool $0300+GTEToolNum
14 <<<

```



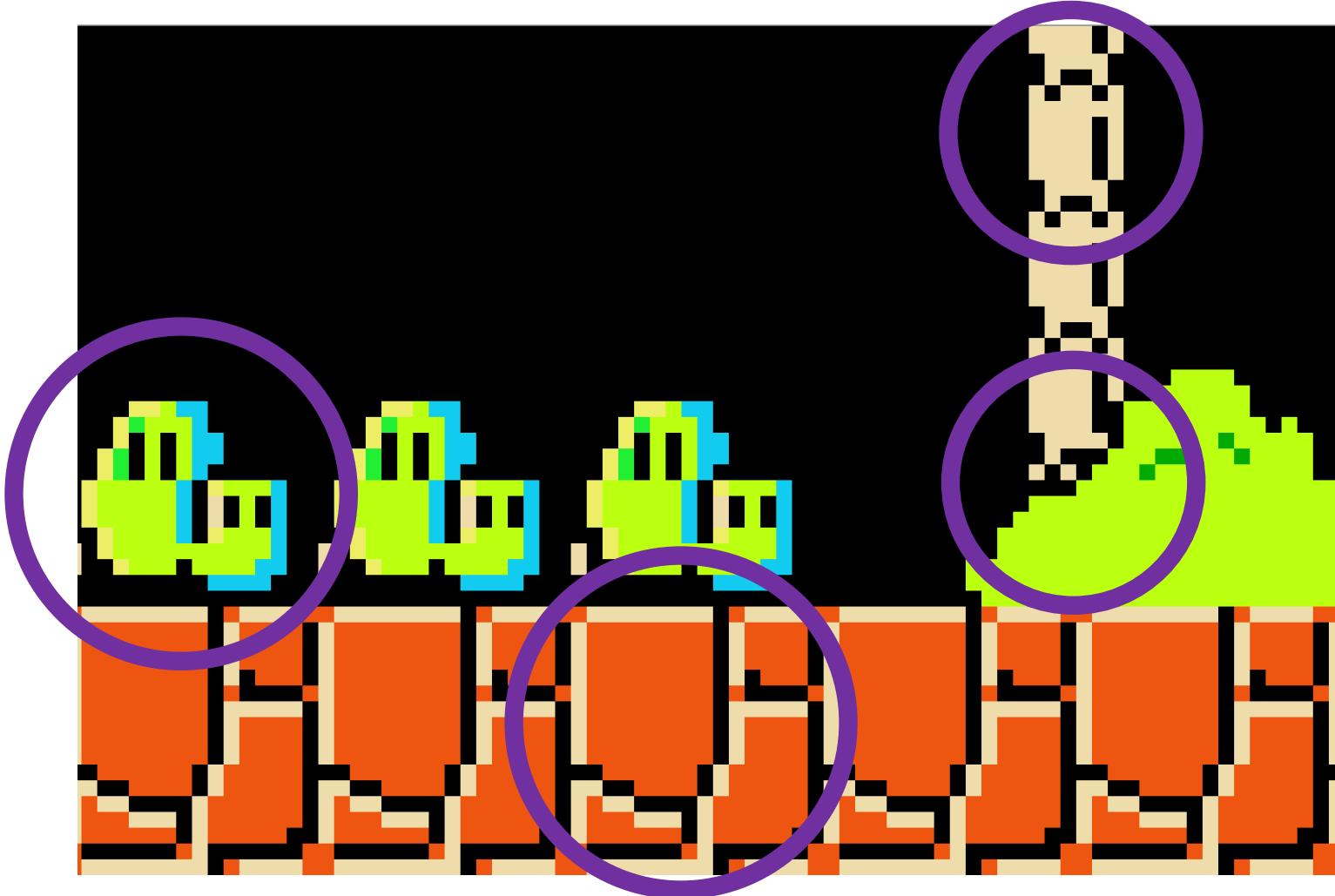
# Backgrounds / Scrolling

- Maintain 208 lines of 82 3-byte instructions each
- 64kb per 16 lines (832kb!!)

```
jmp    enter   ; Patch based on horizontal scroll
:loop  pea     $0000 ← ; 82 instructions (screen width + 1 tile)
       pea     $0000
       pea     $0000
       ...
       pea     $0000
bra    exit    ; Patch based on playfield width
       pea     $0000
       jmp    :loop  ; Wrap-around
exit
```

The assembly code shows a loop structure for rendering backgrounds. It starts at the label ':loop', which is preceded by a jump instruction ('jmp'). Inside the loop, there is an 'enter' instruction followed by three 'pea' instructions. A bracket groups the first 'pea' instruction and its value '\$0000'. An arrow points from the start of the ':loop' label to this bracketed group, indicating that the value '\$0000' is a patch point for horizontal scrolling. The loop continues with several more 'pea' and blank lines, followed by a branch instruction ('bra') to the 'exit' label. This branch is preceded by a 'pea' instruction and its value '\$0000'. Another bracket groups the 'pea' instruction and its value '\$0000', with an arrow pointing from the start of the 'exit' label to it, indicating it is a patch point for playfield width. Finally, a 'jmp' instruction returns control to the start of the loop (':loop').

# Backgrounds / Scrolling





# Dynamic Tiles

- Reads data from direct page
- Direct page set per line

```
lda    #DP_ADDR ; Fixed per line
tcd

...
jmp    enter
:loop  pea    $0000
       lda    $00,x      ; Load data from direct page
       pha                    ; and push onto stack
       ...
       pea    $0000
       jmp    :loop        ; Wrap-around
exit
```



# Secondary Background

- Reads data from separate bank
- Y-register used to set secondary scroll position

```
ldy    #ADDR      ; Patched when vertical origin changes
...
:jmp   enter
:loop  pea  $0000
       lda  ($00),y ; Load data from data bank
       pha            ; and push onto stack
...
       pea  $0000
       jmp  :loop    ; Wrap-around
exit
```



# Secondary Background Cont.

- What if the foreground does not have a full word of transparency?

```
:loop    jmp   enter
        pea   $0000
        jmp   snippet_00 ; Jump to the snippet handler...
        pea   $0000      ; ...and return here
        pea   $0000
        ...
        pea   $0000
```



# Secondary Background Cont.

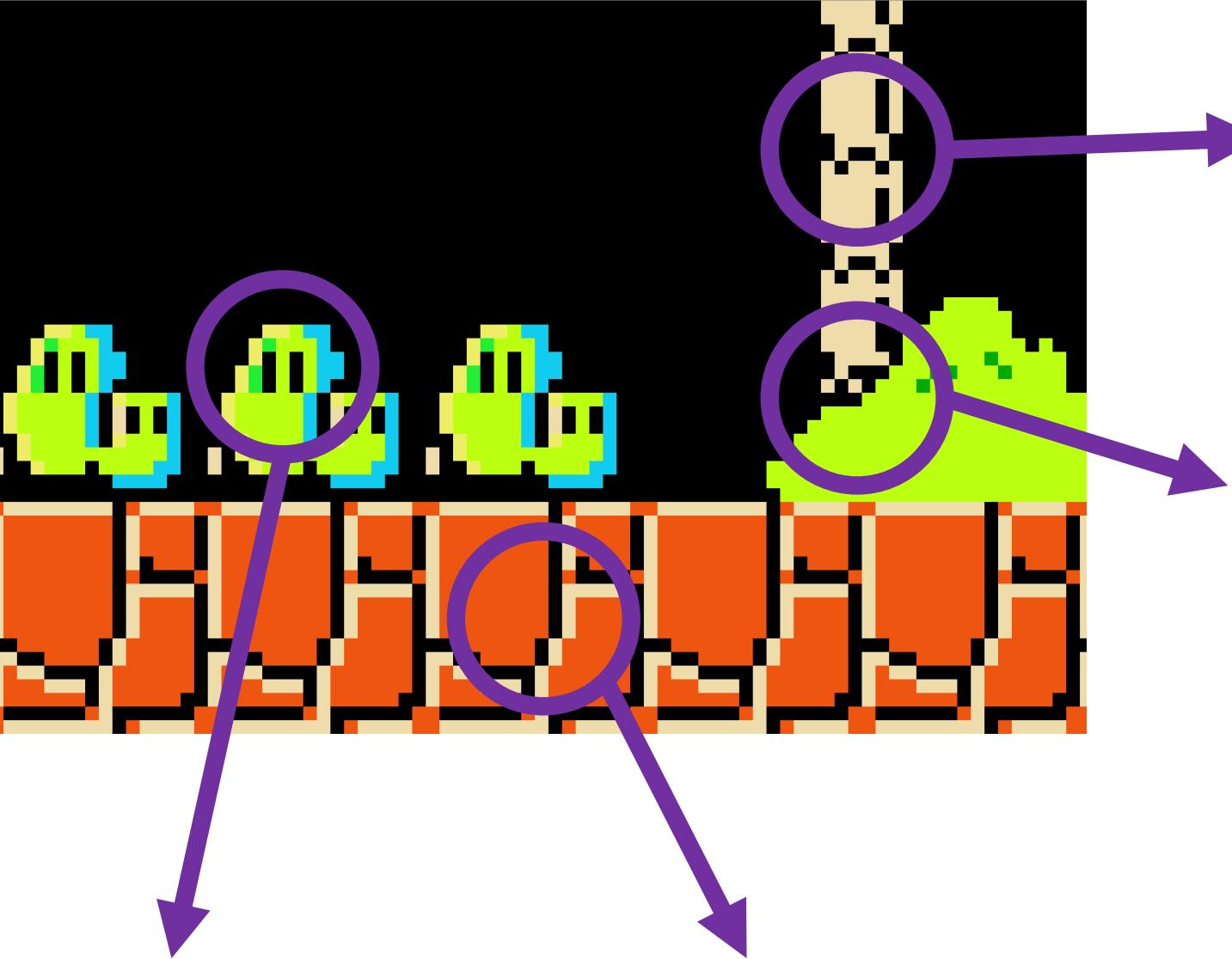
- What if the foreground does not have a full word of transparency?

```
snippet_00    lda    (00),y
                and    #MASK
                ora    #DATA
                pha
                jmp    rtn_00
```

Without Snippets



With Snippets



LDA 00, x  
PHA

PEA #DATA

LDA (00), y  
PHA

RTN\_00: JMP SNP\_00  
RTN\_01: ...

SNP\_00: LDA (00), y  
AND #MASK  
ORA #DATA  
PHA  
JMP RTN\_01

Dirty Tile  
Queue



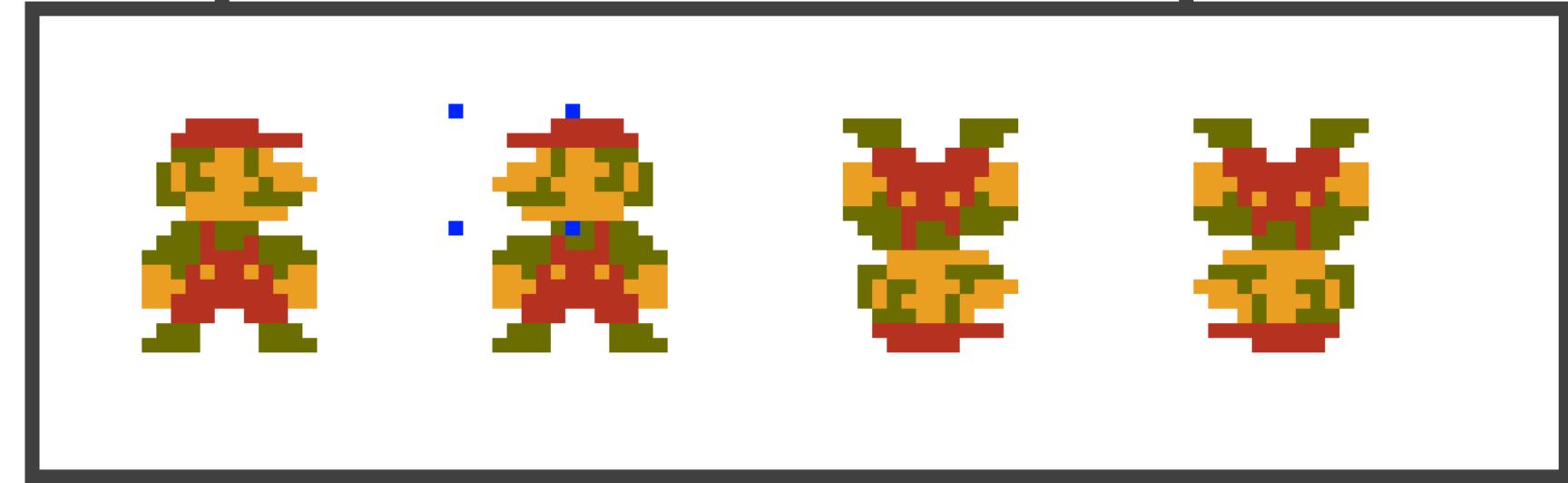
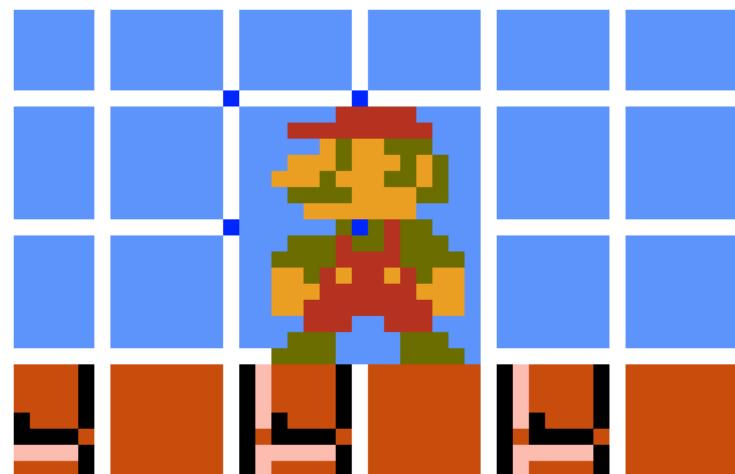
Draw Tile



Blit



sprite stamp





# Future Work

- Performance
  - Sparse Tile Copies
  - Fixed Snippets
  - Per-tile render mode selection
- Features
  - Dynamic V-Sync
  - Expand scripting subsystem
  - Sprite priority flag
  - Expand BG1 rotation range to >64KB
  - Internal size of 320x240 to align with consoles
  - Separate sprite tileset?
  - More sophisticated dynamic tile management?