NES Labyrinth Game

Spring 2024 - CIIC 4082: Computer Architecture II

By: Alanis Negroni & Axel Orta

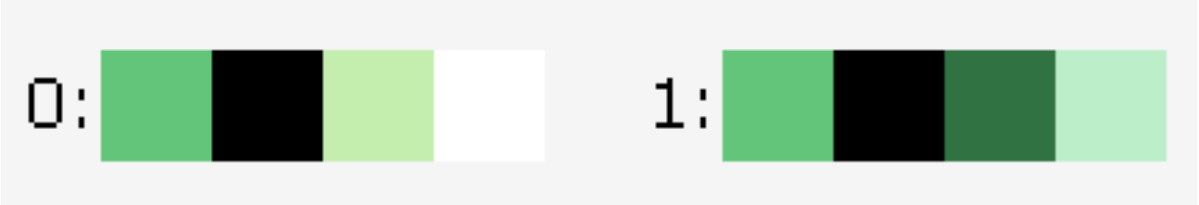
Links

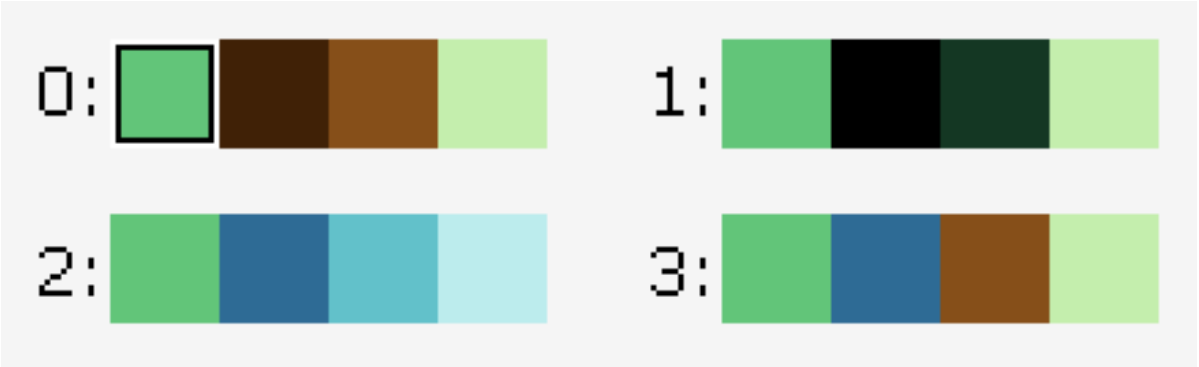
- Github Repository
- YouTube Playlist
- Task 1: Github, Video
- Task 2: Github, Video
- Task 3: Github, Video

Task 1 Static Sprite and Background Pattern Definition and Rendering

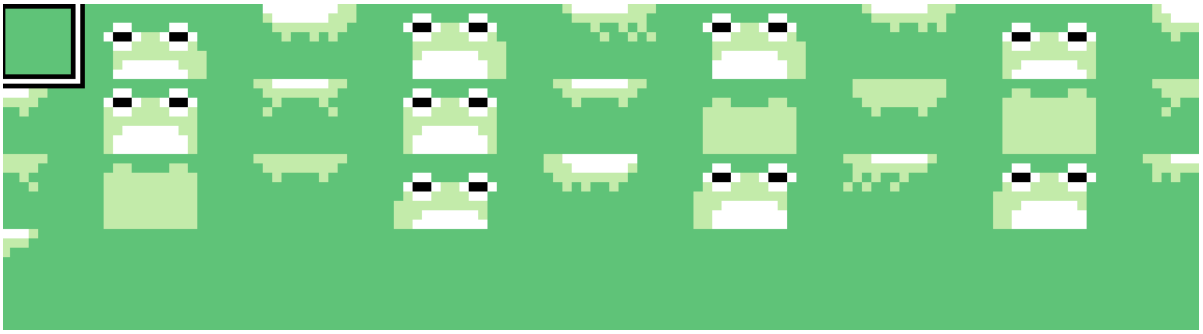
Graphics

Palettes





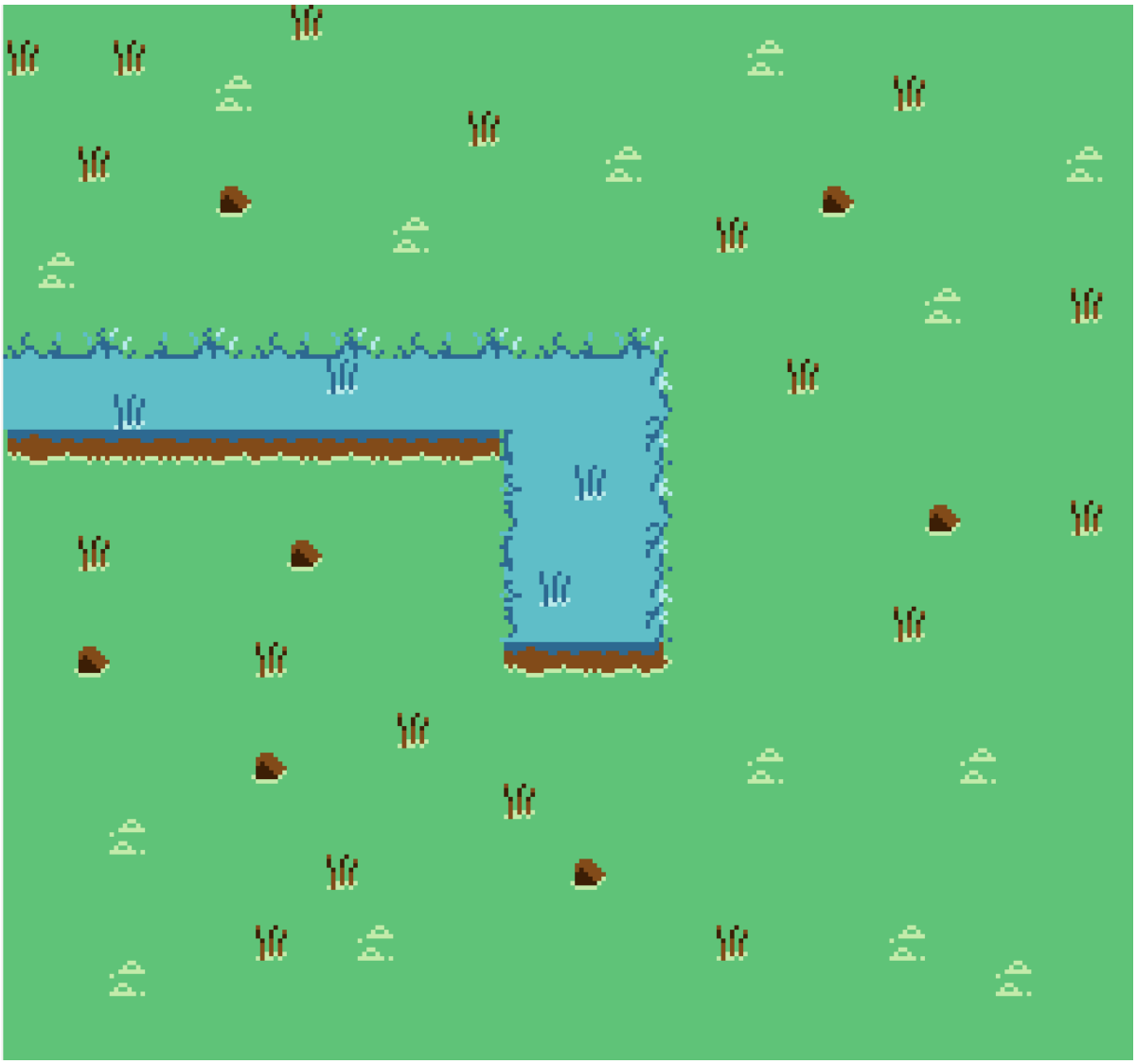
Sprite Tiles



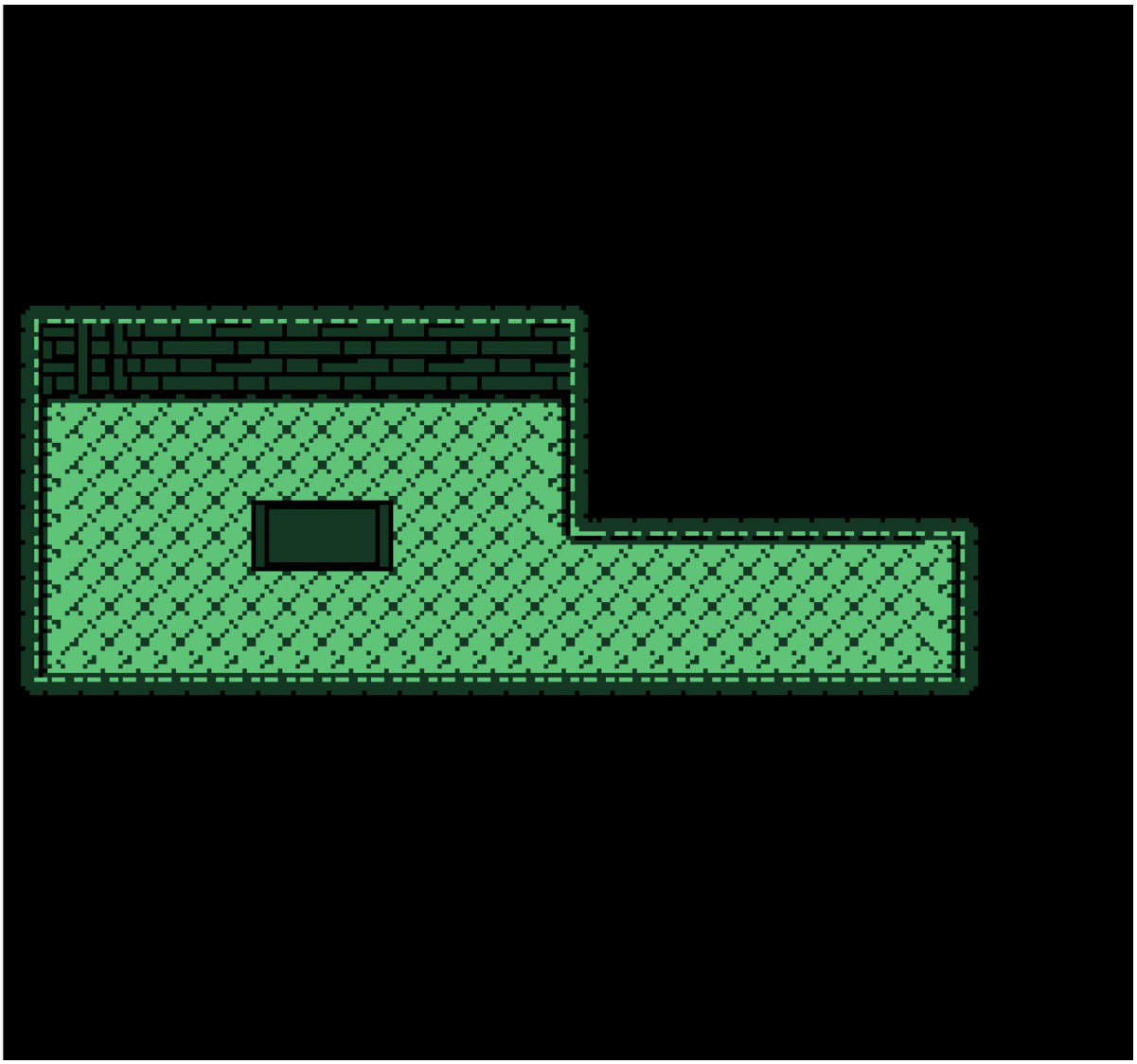
Frog

Background Tiles

Background 1Example



Background 2 Example



Theory

Common Name	Address
PPUCTRL	$2000
PPUMASK	$2001
PPUSTATUS	$2002
OAMADDR	$2003
OAMDATA	$2004
PPUSCROLL	$2005
PPUADDR	$2006
PPUDATA	$2007
OAMDMA	$4014

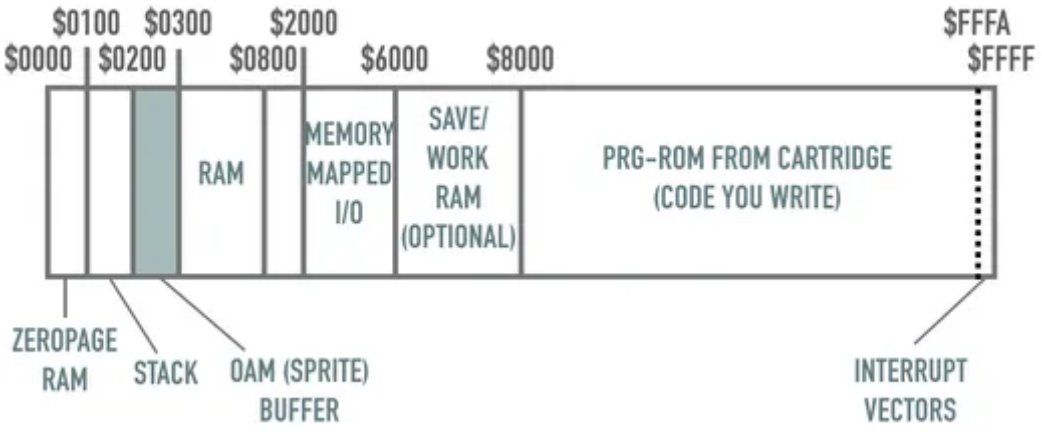
Palettes

```
LDX $2002             ; Load PPU Status into X

LDX #$3f              ; Load 3f into X
STX $2006             ; Store 3f in PPU Address
LDX #$00              ; Load 00 into X
STX $2006             ; Store 00 in PPU Address, such that we have 3f00 (the first color

; WRITE PALETTES ------------------------------------------------------------+
load_palettes:
    LDA palettes, X       ; Load palettes into X
    STA $2007             ; Store X into PPU data
    INX                   ; Increase X
    CPX #$18              ; Compare X. If X > 24 (6 patterns tables)
    BNE load_palettes     ;
```

$00	$01	$02	$03	$04	$05	$06	$07	$08	$09	$0A	$0B	$0C	$0F
$10	$11	$12	$13	$14	$15	$16	$17	$18	$19	$1A	$1B	$1C	$0F
$20	$21	$22	$23	$24	$25	$26	$27	$28	$29	$2A	$2B	$2C	$2D
$30	$31	$32	$33	$34	$35	$36	$37	$38	$39	$3A	$3B	$3C	$3D

```
.segment "RODATA"               ; read-only data
    palettes:
        ; BACKGROUND PALETTE ------------------------------------------------+
        .byte $2B, $07, $17, $3A    ; green brown, brown, light green
        .byte $2B, $0F, $0B, $3A    ; green, dark green, light green
        .byte $2B, $1C, $2C, $3C    ; green blue, blue, light blue
        .byte $2B, $1C, $17, $3A    ; green blue, brown, light green
        ; SPRITE PALETTE ------------------------------------------------------+
        .byte $2B, $0f, $3A, $20    ; green, black, green, white
        .byte $2B, $0f, $1B, $3B    ; green, black, green, green
```

Sprites

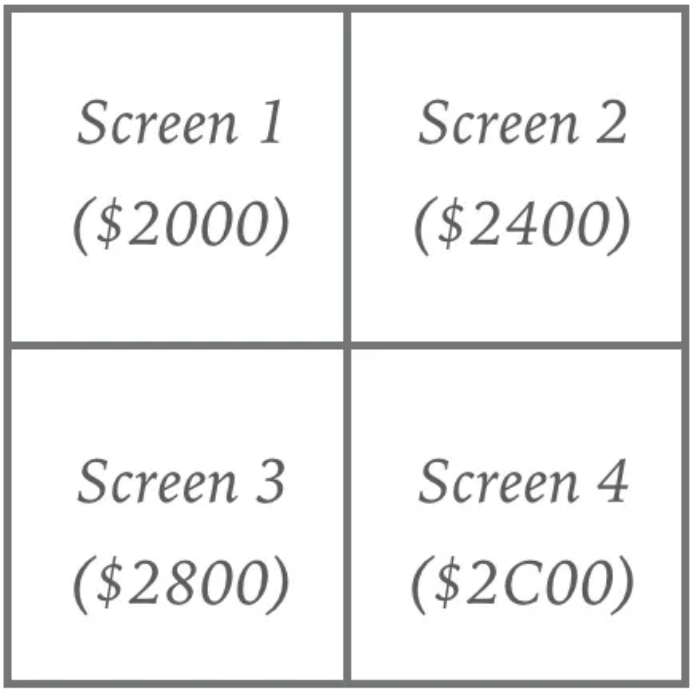


```
; WRITE SPRITE DATA ------------------------------------------------------------+
LDX #$00                ; Set X to 0
load_sprites:           ; Iterate through the sprites to draw them
    LDA sprites,X       ; Load the sprites into X
    STA $0200,X         ; Store X into 0200
    INX                 ; Increase X
    CPX #$CA            ; Compare X. If X > 192 (12 16bit sprites) stop the loop
    BNE load_sprites
```

```
.segment "RODATA"               ; read-only data      You, 3 days ago • Task 1: Sprites and Backgrounds Implemented
   sprites:
   ; LEFT ------------------------------------------------------------------------+
      ; STANDING
       .byte $40, $01, 00, $60     ; y = 40, tile number = 01, Special attribute flags - palette = 00, x = 60
       .byte $40, $02, 00, $68     ; y = 40, tile number = 02, Special attribute flags - palette = 00, x = 68
       .byte $48, $03, 01, $60     ; y = 48, tile number = 03, Special attribute flags - palette = 01, x = 60
       .byte $48, $04, 01, $68     ; y = 48, tile number = 04, Special attribute flags - palette = 01, x = 68
      ; JUMPING 1
       .byte $40, $05, 00, $70     ; y = 40, tile number = 05, Special attribute flags - palette = 00, x = 70
       .byte $40, $06, 00, $78     ; y = 40, tile number = 06, Special attribute flags - palette = 00, x = 78
       .byte $48, $07, 01, $70     ; y = 48, tile number = 07, Special attribute flags - palette = 01, x = 70
       .byte $48, $08, 01, $78     ; y = 48, tile number = 08, Special attribute flags - palette = 01, x = 78
      ; JUMPING 2
       .byte $40, $09, 00, $80     ; y = 40, tile number = 09, Special attribute flags - palette = 00, x = 80
       .byte $40, $0A, 00, $88     ; y = 40, tile number = 0A, Special attribute flags - palette = 00, x = 88
       .byte $48, $0B, 01, $80     ; y = 48, tile number = 0B, Special attribute flags - palette = 01, x = 80
       .byte $48, $0C, 01, $88     ; y = 48, tile number = 0C, Special attribute flags - palette = 01, x = 88
```

```
.segment "CHARS"
.incbin "graphics.chr"
```

Background



```
; BACKGROUND TILES ──────────────────────────────────────────────────────────+
LDA $2002
LDA #$22
STA $2006
LDA #$0C
STA $2006
LDX #$04
STX $2007
```

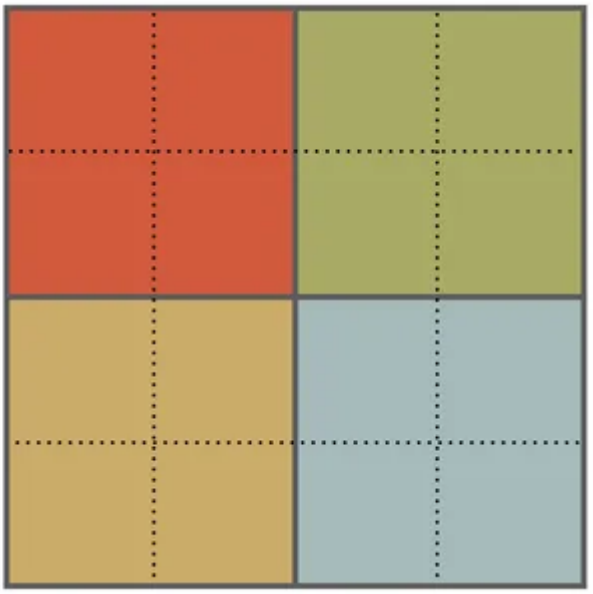
%01100011

Bits 0-1, top left:
%11 = *palette 3*

Bits 2-3, top right:
%00 = *palette 0*

Bits 4-5, bottom left:
%10 = *palette 2*

Bits 6-7, bottom right:
%01 = *palette 1*



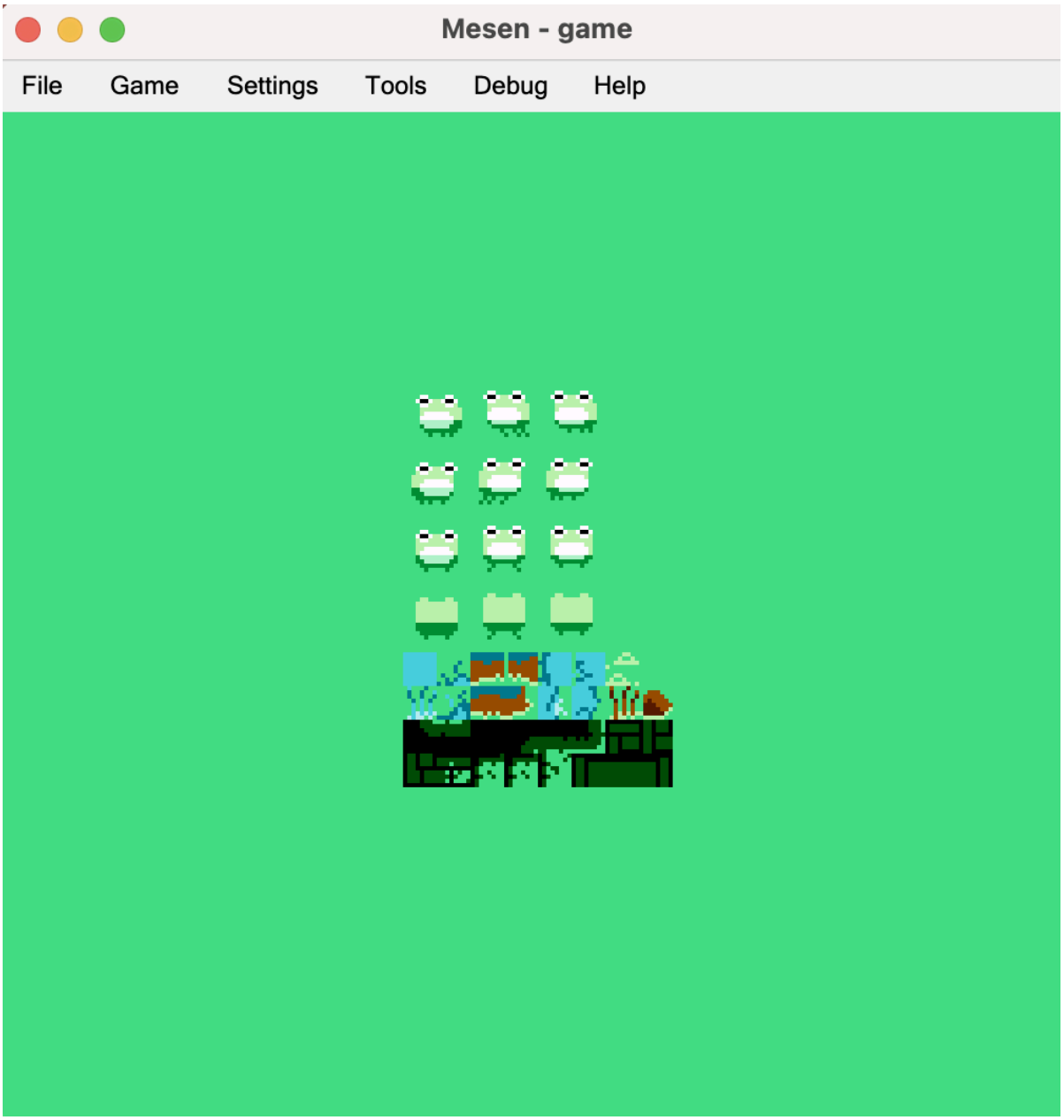
```
; ATTRIBUTE TABLE ──────────────────────────────────────────────────────────+
LDA $2002
LDA #$23
STA $2006
LDA #$E3
STA $2006
LDA #%01011110
STA $2007
```

```
.segment "CHARS"
.incbin "graphics.chr"
```

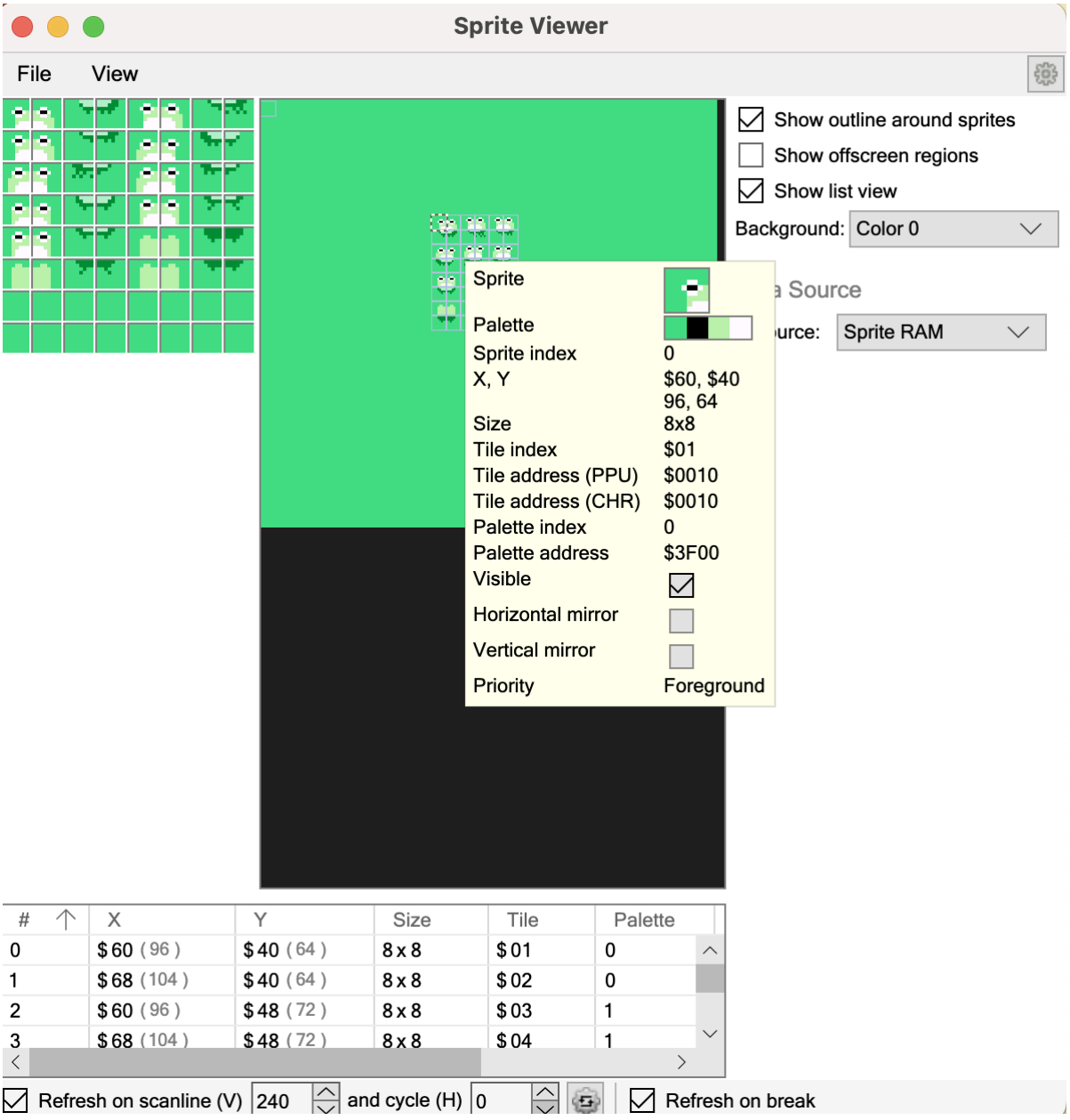
NES Emulator

Commands:

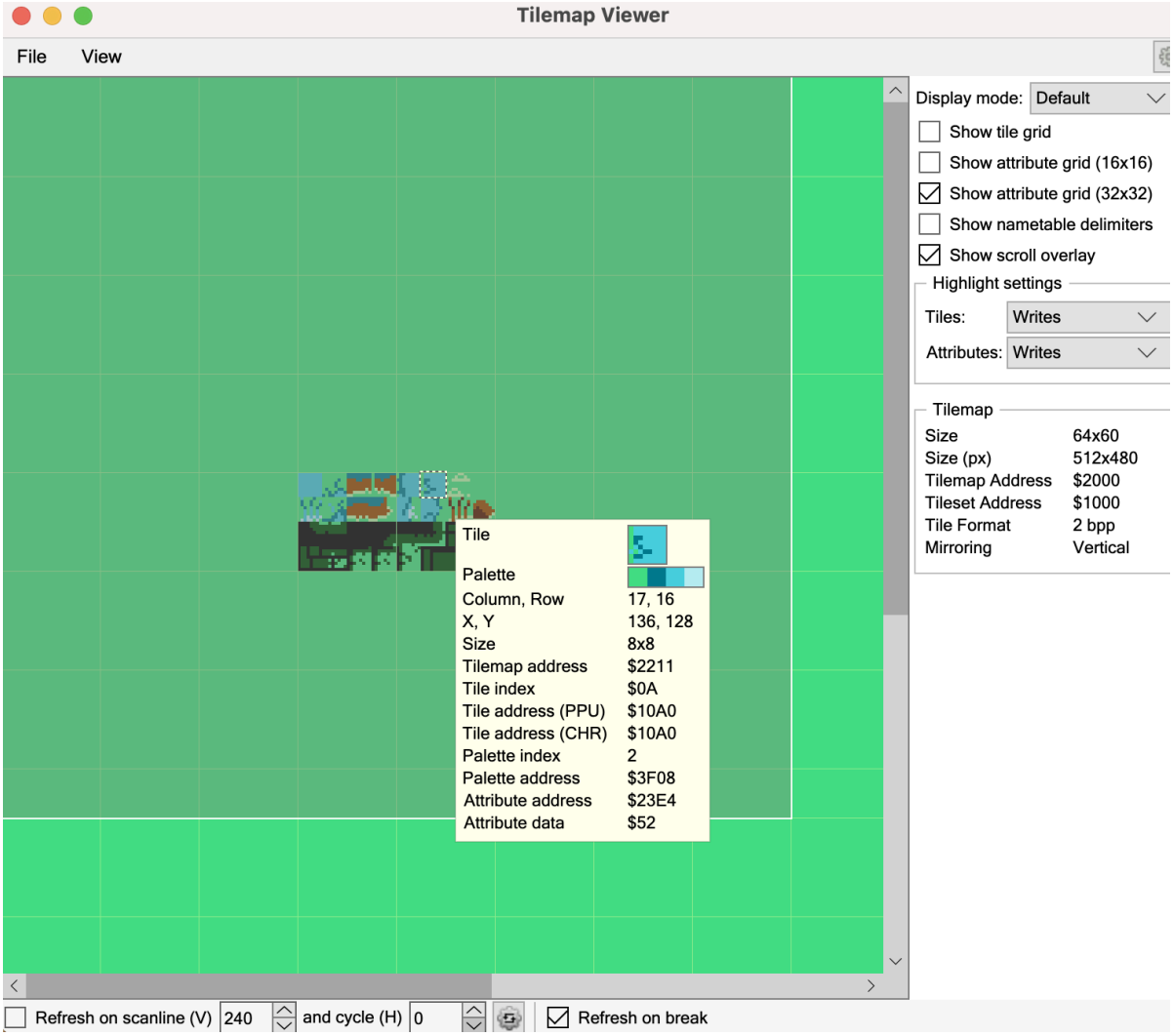
```
cd Task\ 1/
ca65 game.asm && ld65 game.o -t nes -o game.nes
```



Sprites


Sprite Viewer
File
View
Show outline around sprites
Show offscreen regions
Show list view
Background:
Color 0
Source
Sprite RAM
Sprite
Palette
Sprite index 0
X, Y $60, $40
96, 64
Size 8x8
Tile index $01
Tile address (PPU) $0010
Tile address (CHR) $0010
Palette index 0
Palette address $3F00
Visible
Horizontal mirror
Vertical mirror
Priority Foreground
X Y Size Tile Palette
0 $60 (96) $40 (64) 8 x 8 $01 0
1 $68 (104) $40 (64) 8 x 8 $02 0
2 $60 (96) $48 (72) 8 x 8 $03 1
3 $68 (104) $48 (72) 8 x 8 $04 1
Refresh on scanline (V) 240
and cycle (H) 0
Refresh on break

Background


Tilemap Viewer
File
View
Display mode: Default
Show tile grid
Show attribute grid (16x16)
Show attribute grid (32x32)
Show nametable delimiters
Show scroll overlay
Highlight settings
Tiles: Writes
Attributes: Writes
Tilemap
Size 64x60
Size (px) 512x480
Tilemap Address $2000
Tileset Address $1000
Tile Format 2 bpp
Mirroring Vertical
Tile
Palette
Column, Row 17, 16
X, Y 136, 128
Size 8x8
Tilemap address $2211
Tile index $0A
Tile address (PPU) $10A0
Tile address (CHR) $10A0
Palette index 2
Palette address $3F08
Attribute address $23E4
Attribute data $52
Refresh on scanline (V) 240
and cycle (H) 0
Refresh on break

Task 2 Sprite Animation

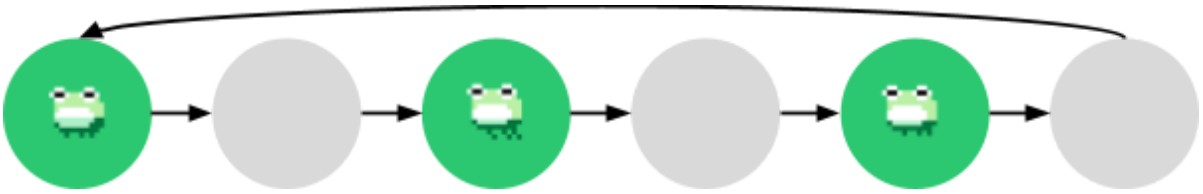
Theory

```
.segment "ZEROPAGE"
    frogX:                  .res 1  ; Reserve 1 byte for the sprite position in X
    frogY:                  .res 1  ; Reserve 1 byte for the sprite position in X
    frogDirection:          .res 1  ; Reserve 1 byte for the sprite direction
    animationState:         .res 1  ; Reserve 1 byte for the animation state for the FSM
    tick:                   .res 1  ; Reserve 1 byte for the a tick counter (frame counter)
    frogOffSet:             .res 1  ; Reserve 1 byte for the frog off-set, to be able to add multiple fr
    directionOffSet:        .res 1  ; Reserve 1 byte for the frog direction off-set, to able to have mul
```

Animation States

```
.proc update    ; ————————————————————+
    LDA tick    ; Load tick
    CLC         ; Clear carry flag
    ADC #$01    ; Add 1 to tick
    STA tick    ; Store new tick
.endproc
```

```
   LDA tick                 ; Load current tick
   CMP #$1E
   BEQ changeAnimation      ; Go to changeAnimation if tick is 30
   JMP continue             ; Else continue (go to drawFrog)
```



```
changeAnimation:
    LDA #$00
    STA tick                ; Set tick to 0
    LDA animationState
    CLC
    ADC #$01
    STA animationState      ; Add 1 to the animation state
    LDA animationState
    CMP #$01
    BEQ stateIncrement      ; If animation state is 1 go to state increment
    CMP #$03
    BEQ stateIncrement      ; If animation state is 3 go to state increment
    CMP #$05
    BEQ stateDecrement      ; If animation state is 5 go to state decrement
    JMP continue
```

```
stateIncrement:
    LDA directionOffSet
    CLC
    ADC #$04
    STA directionOffSet ; Increase direction offset by four (1 tile)
    LDA animationState
    CLC
    ADC #$01
    STA animationState  ; Add 1 to animation state (transition state, with for another tick)
    JMP continue

stateDecrement:
    LDA directionOffSet
    CLC
    SBC #$07
    STA directionOffSet ; Set direction offset to original direction
    LDA #$00
    STA animationState  ; Set state to first (original state)
    JMP continue
```

Multiple Frogs

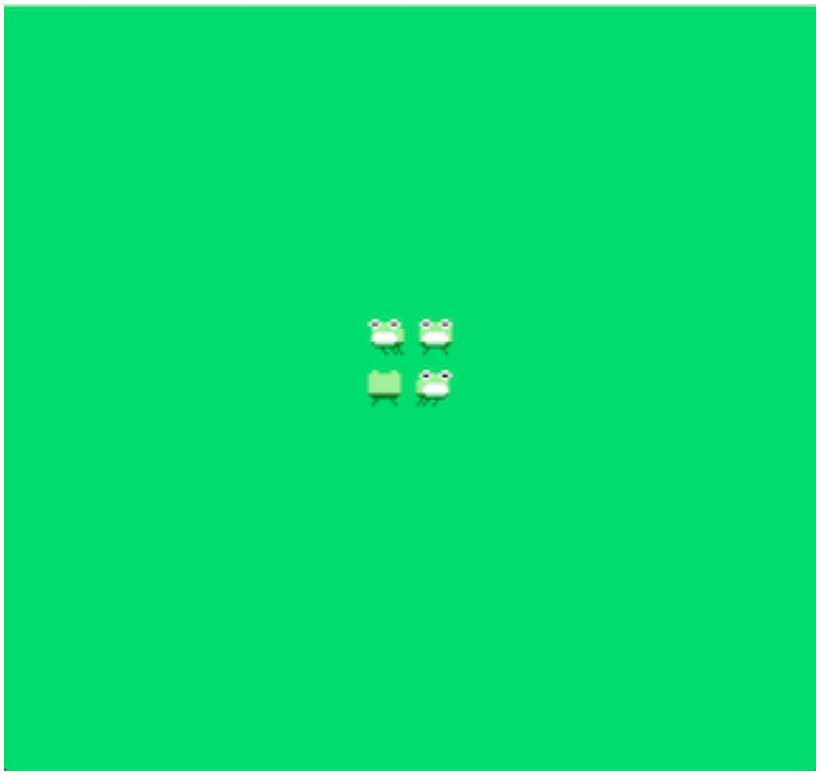
```
 LDA frogDirection
 CLC
 ADC directionOffSet
 STA frogDirection       ; Add direction offset to the frog direction

 LDX frogOffSet          ; Load frog offset to X

 ; DRAW UPPER LEFT TILE
 LDA frogY
 STA $0200, X
 LDA frogDirection
 STA $0201, X
 LDA #$00
 STA $0202, X
 LDA frogX
 STA $0203, X
```

```
; DRAW LOWER RIGHT TILE
LDA frogY
CLC
ADC #$08
STA $020C, X
LDA frogDirection
CLC
ADC #$03
STA $020D, X
LDA #$01
STA $020E, X
LDA frogX
CLC
ADC #$08
STA $020F, X

LDA frogOffSet
CLC
ADC #$10
STA frogOffSet          ; Increase frog offset by 16
```

NES Emulator

Commands:

```
cd Task\ 2/
ca65 game.asm && ld65 game.o -t nes -o game.nes
```



Task 3 Controlling the Character

Theory

Controllers


```
read_buttons:
    LDA $4016       ; Read button state
    LSR             ; Shift right, moving the button's state into carry
    ROL temp        ; Rotate Left through Carry to move button state into temp
    DEY
    BNE read_buttons

    ; Check Right (bit 7 of temp)
    LDA temp
    AND #%10000000  ; Isolate Right button
    BEQ check_left  ; If 0, button not pressed, check next
    LDA frogX
        ; Boundaries break animations when walked against
    ; CMP #$EF        ; 240 - Maximum X value before moving off-screen to the right
    ; BCS update_done ; Skip increment if frogX ≥ #$EF
    INC frogX       ; Move right
    LDA #$25
    STA frogDirection ; Set frogDirection to 25 (Right)
        LDA #$01
        STA hasMoved ; Set hasMoved to 1
    JMP update_done
```

```
.proc ClearSprites
    LDX #$00            ; Start with the first sprite
clear_loop:
    LDA #$FF            ; Load A with $FF, a Y position off-screen
    STA $0200, X        ; Set the sprite's Y position off-screen, taking $0200 as the start of OAM
    INX                 ; Move to the next byte in OAM
    INX                 ; Skip over the attribute byte
    INX                 ; Skip over the X position byte
    INX                 ; Advance to the next sprite's Y position
    CPX #$00            ; Check if X has rolled over, indicating all 64 sprites were processed
    BNE clear_loop
    RTS                 ; Return from subroutine
.endproc
```

NES Emulator

Commands:

```
cd Task\ 3/
ca65 game.asm && ld65 game.o -t nes -o game.nes
```