Atari 2600 Homebrew

Darrell Spice, Jr.

What is Homebrew?

- Games (or other software) made by hobbyists for platforms that are not typically end user programmable

- Over 100 have been released for the Atari 2600

- AtariAge has 80+ homebrew 2600 titles available, only Atari had a larger catalog of 2600 games

My Homebrew games

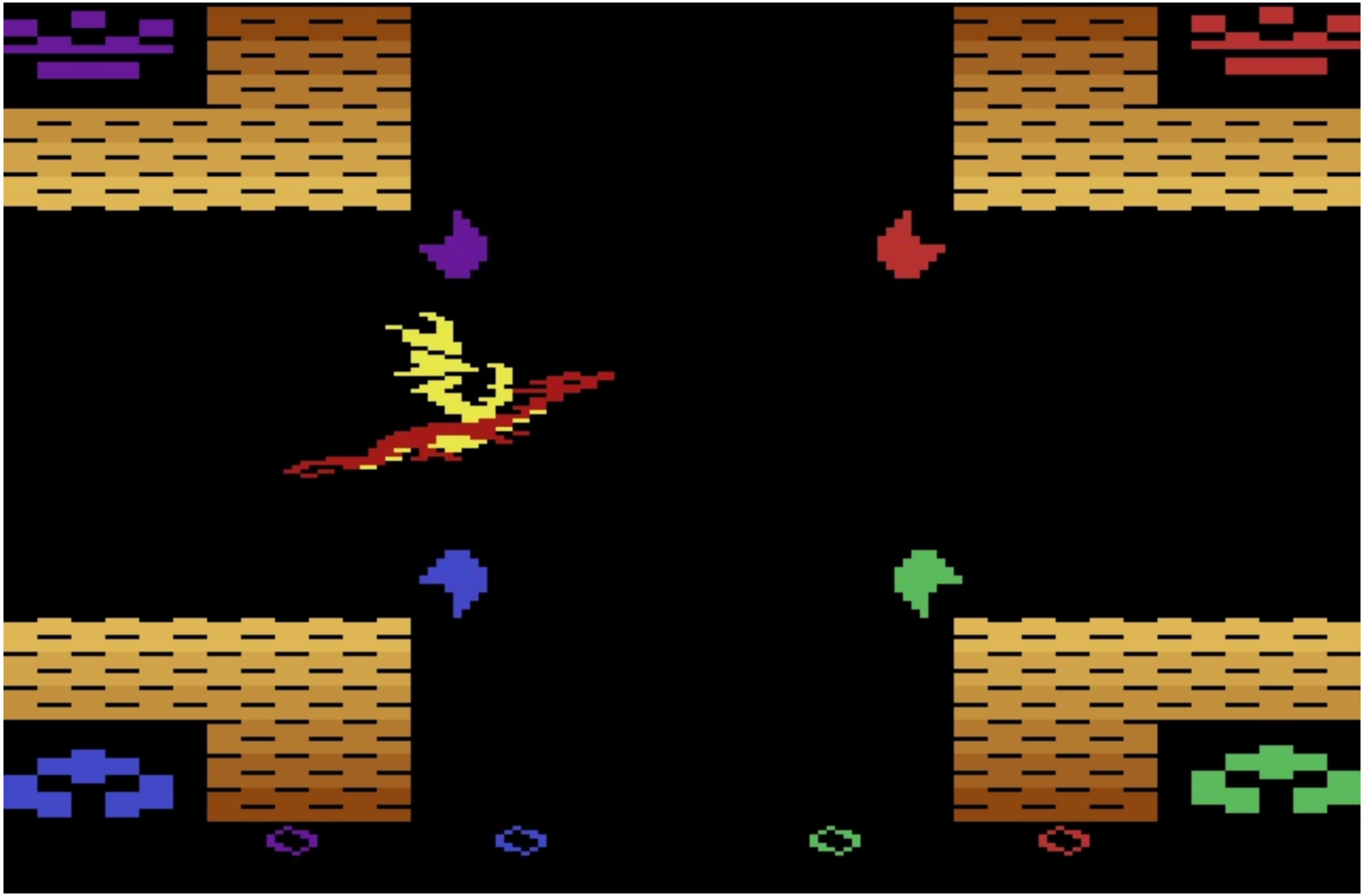
Finished

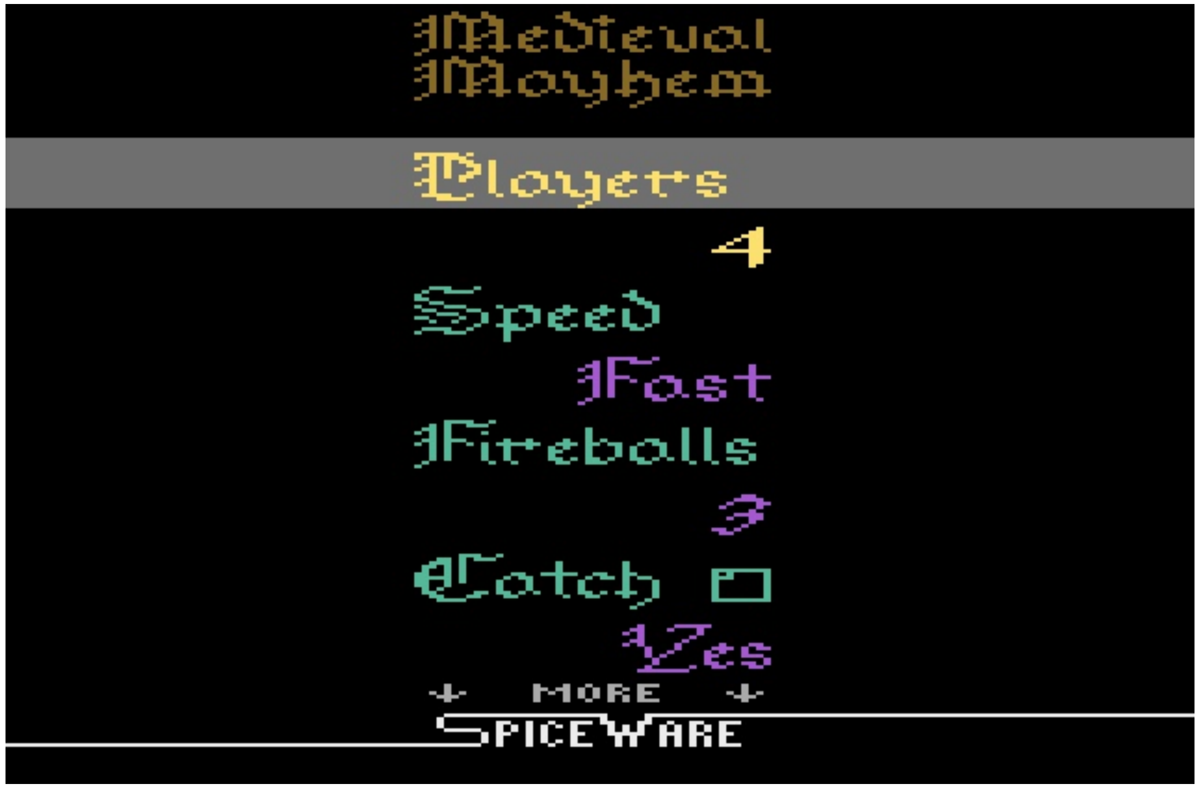
- Medieval Mayhem
- Space Rocks
- Stay Frosty
- Stay Frosty 2

WIP

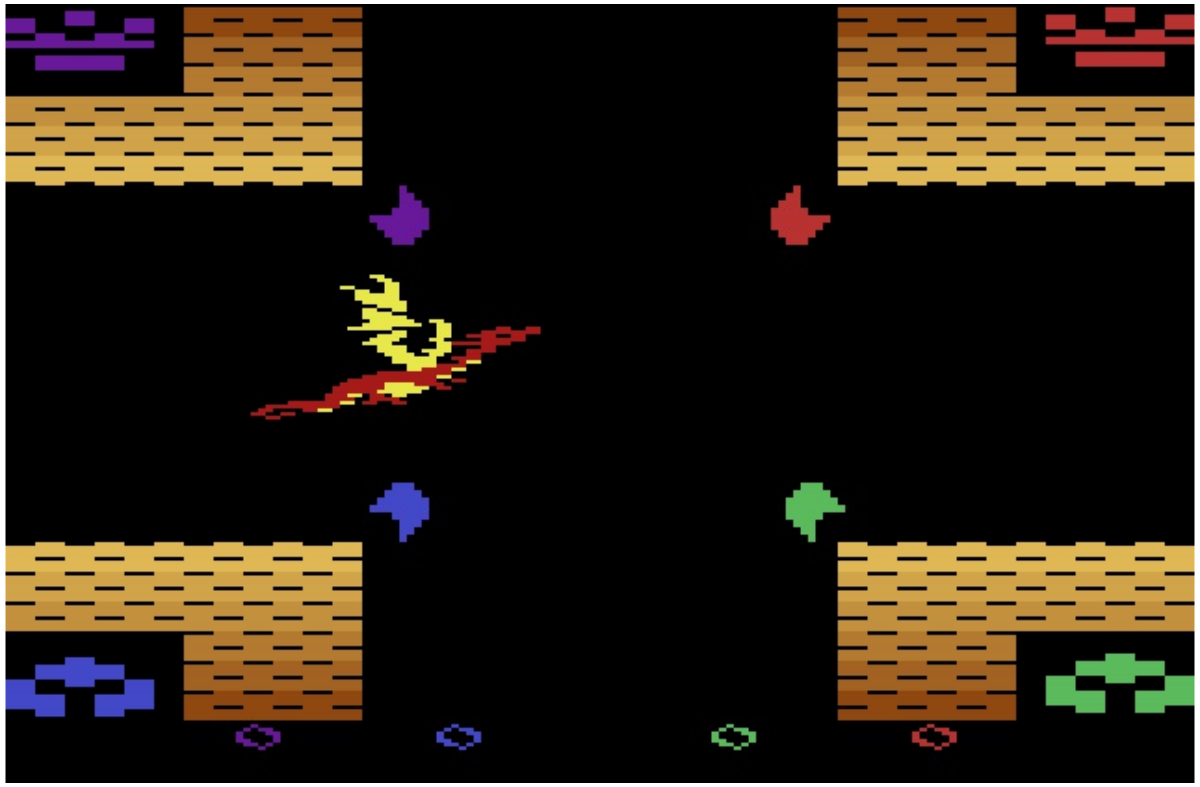
- Frantic
- Timmy

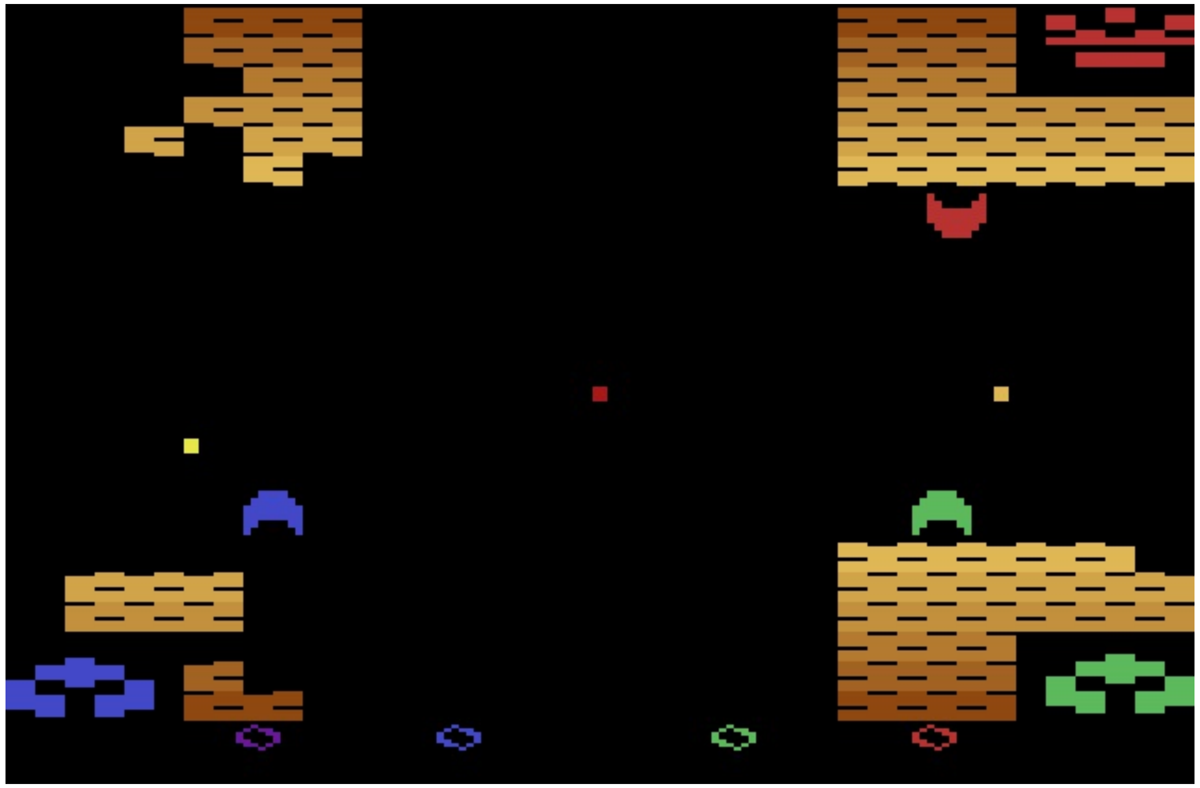
Medieval Mayhem

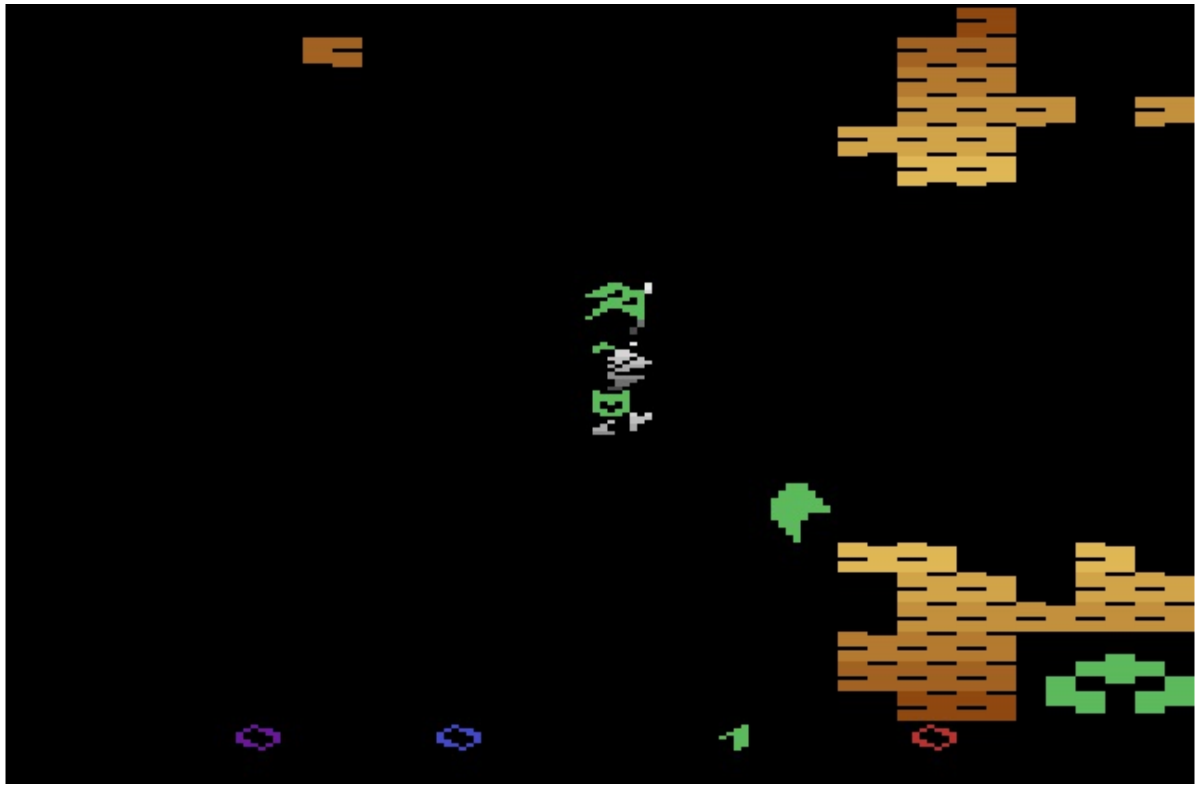




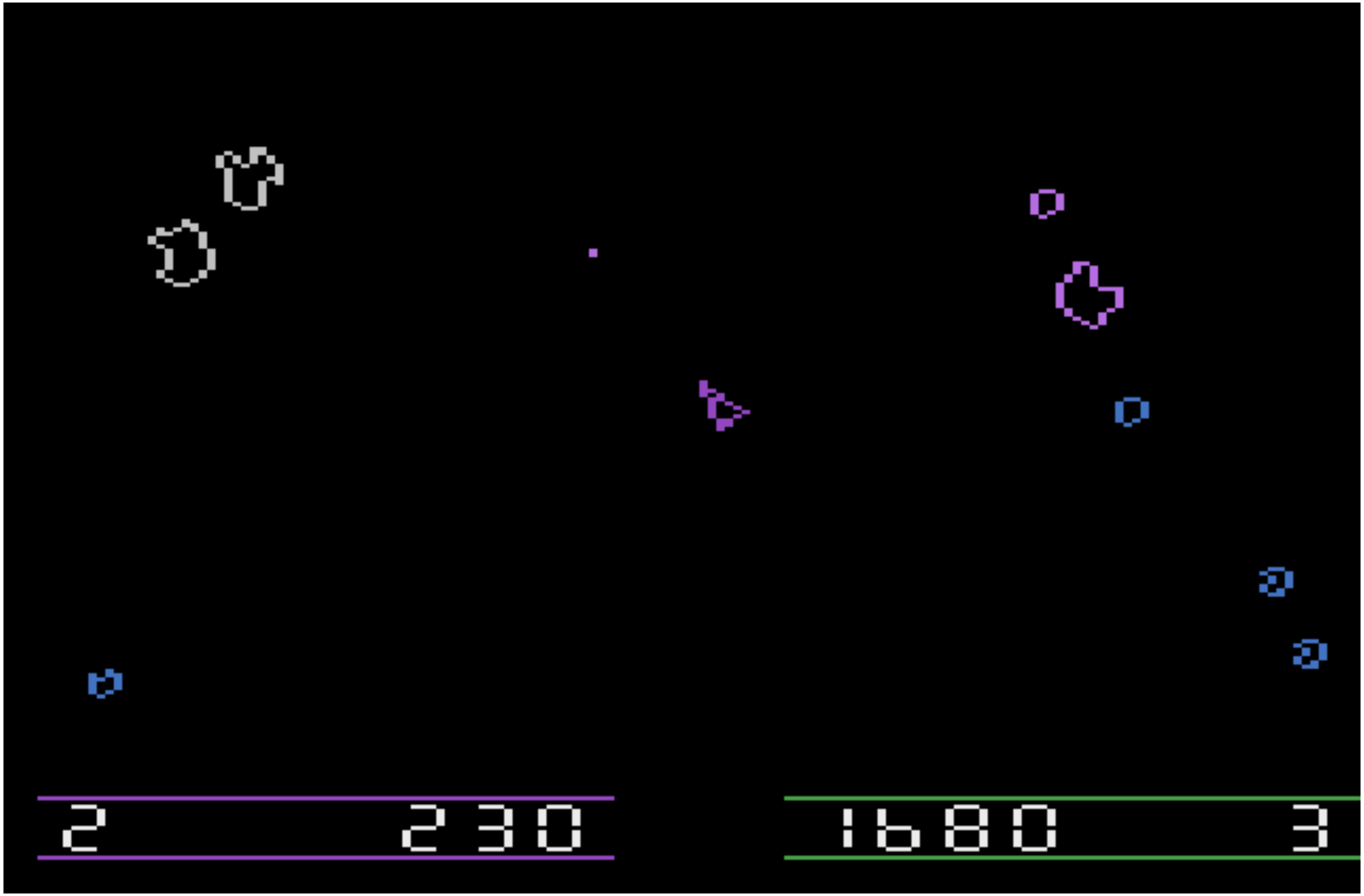
Players
4
Speed
Fast
Fireballs
3
Catch □
Yes

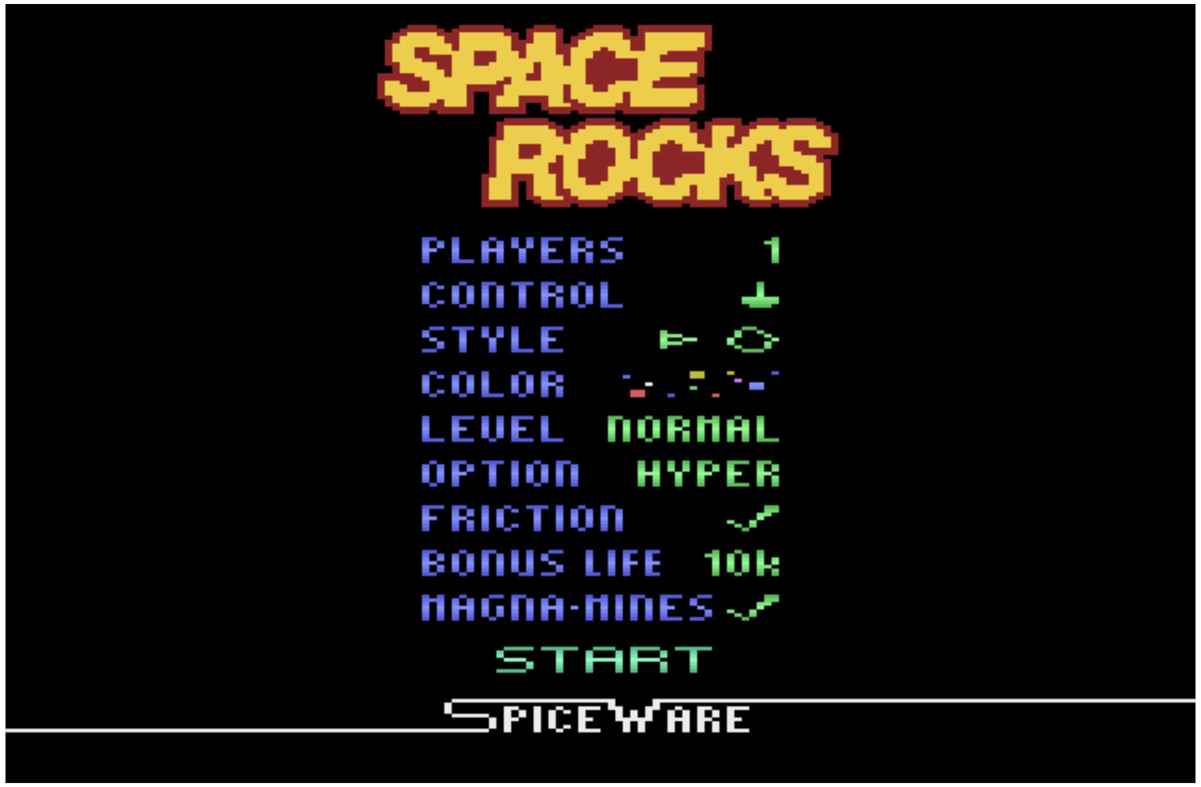






Space Rocks





SPACE ROCKS

PLAYERS 1

CONTROL

STYLE

COLOR

LEVEL NORMAL

OPTION HYPER

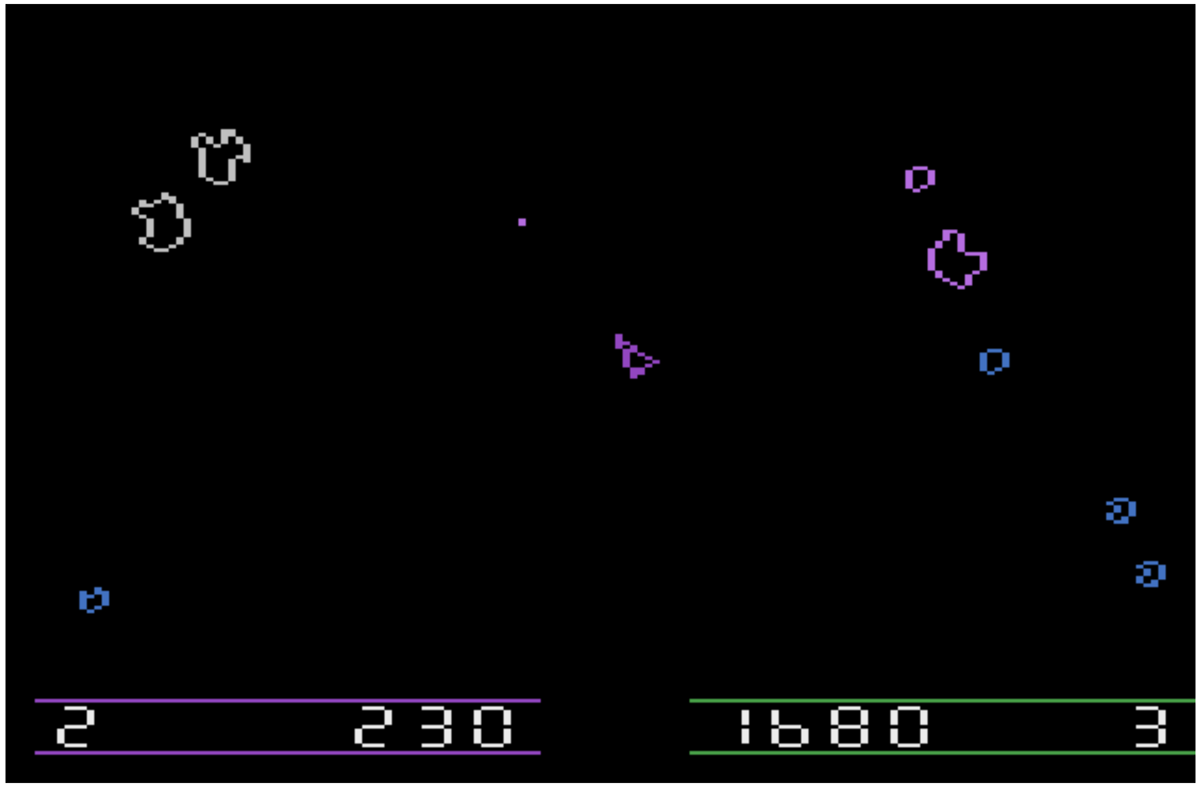
FRICTION ✓

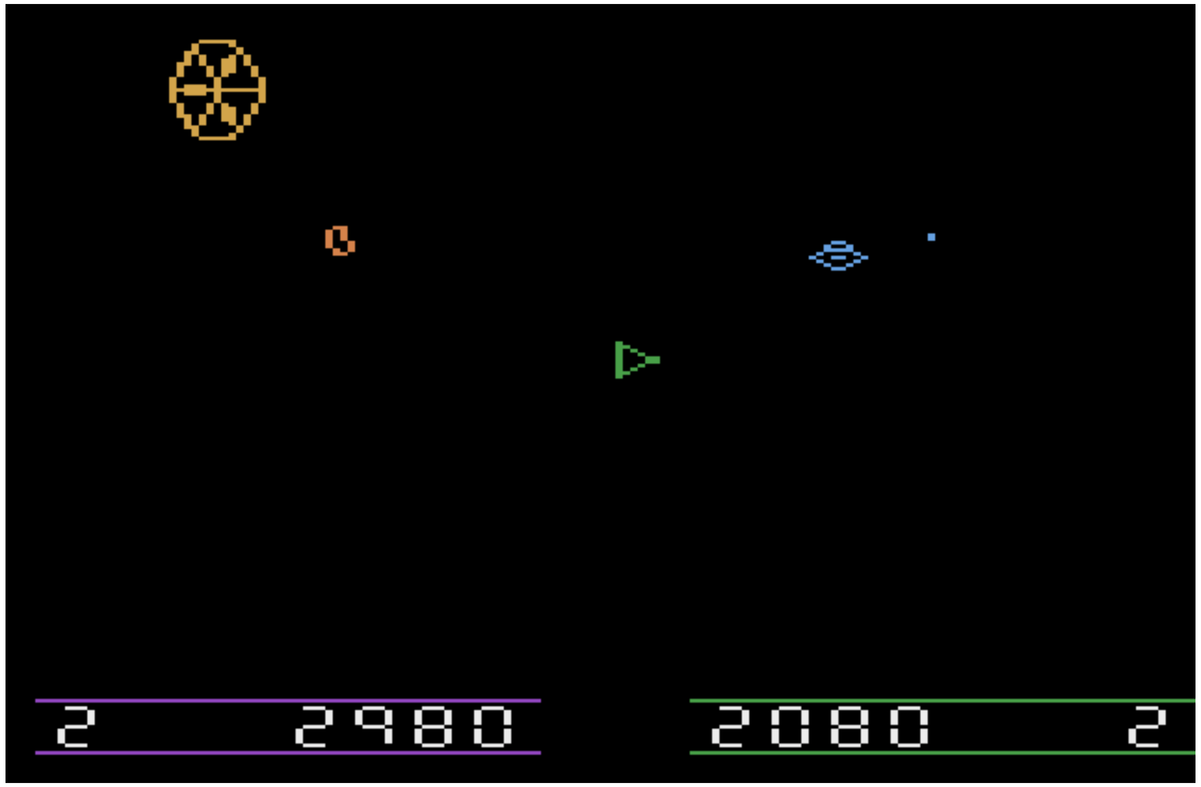
BONUS LIFE 10k

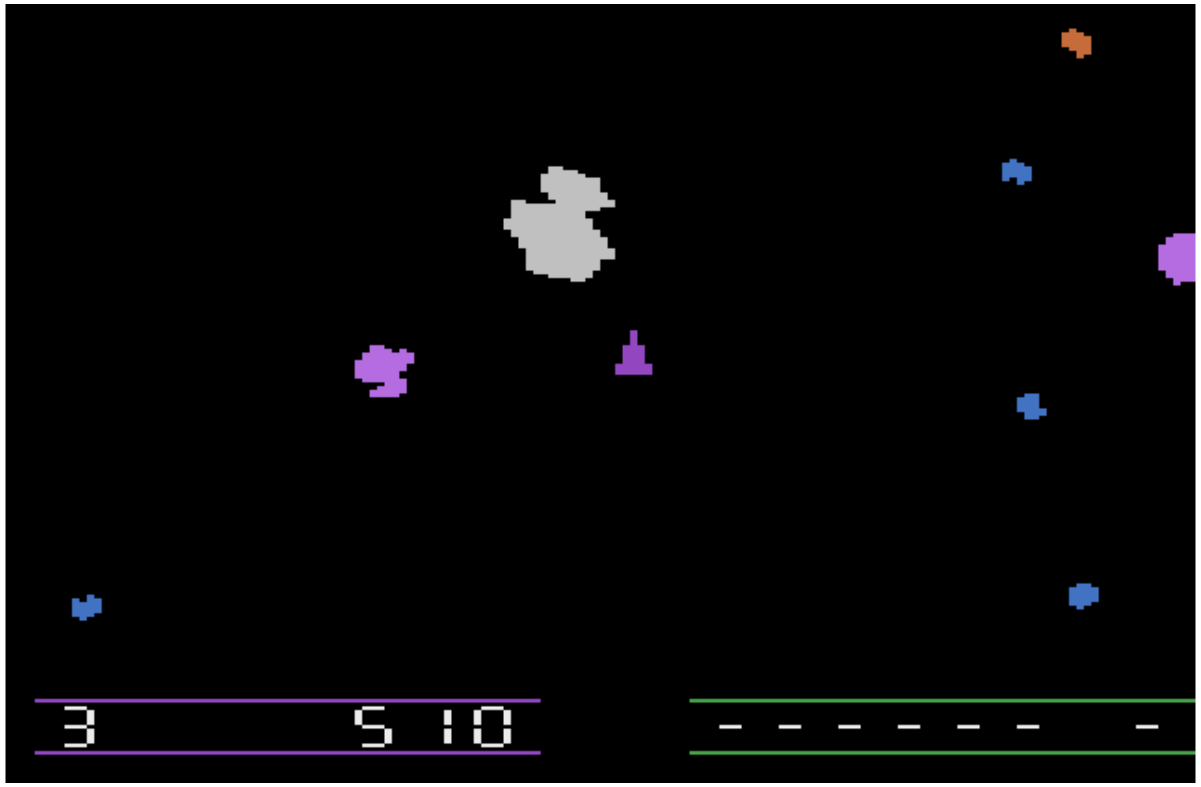
MAGNA-MINES ✓

START

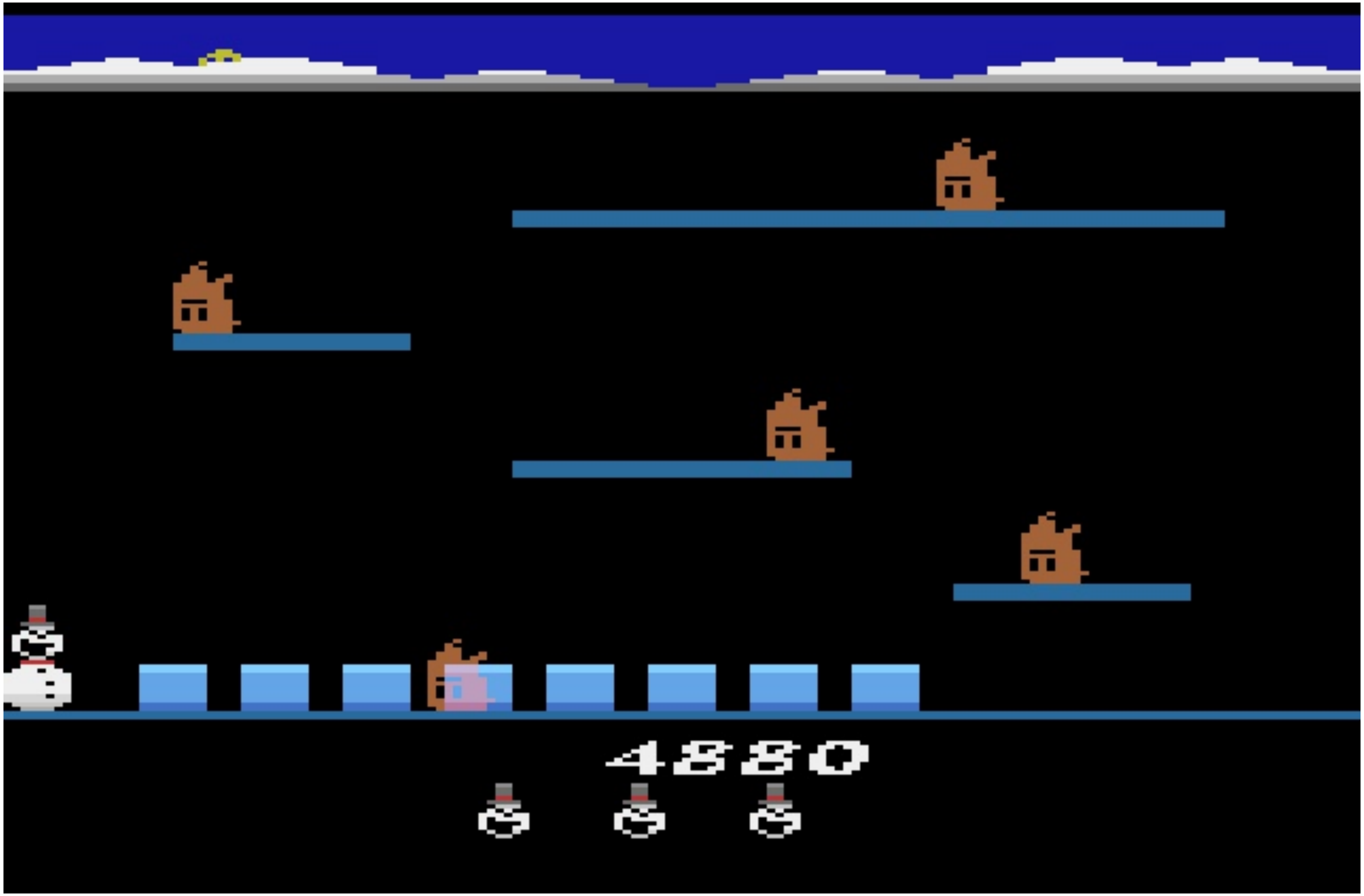
SPICEWARE

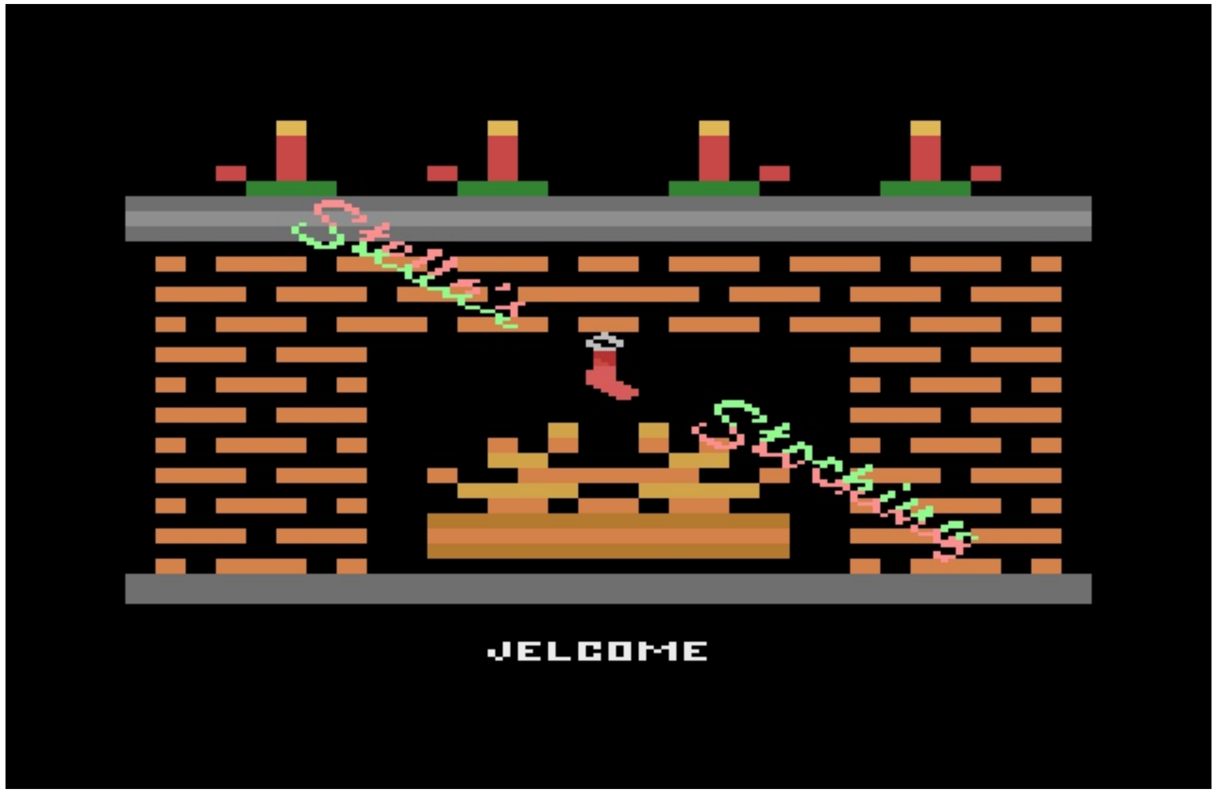


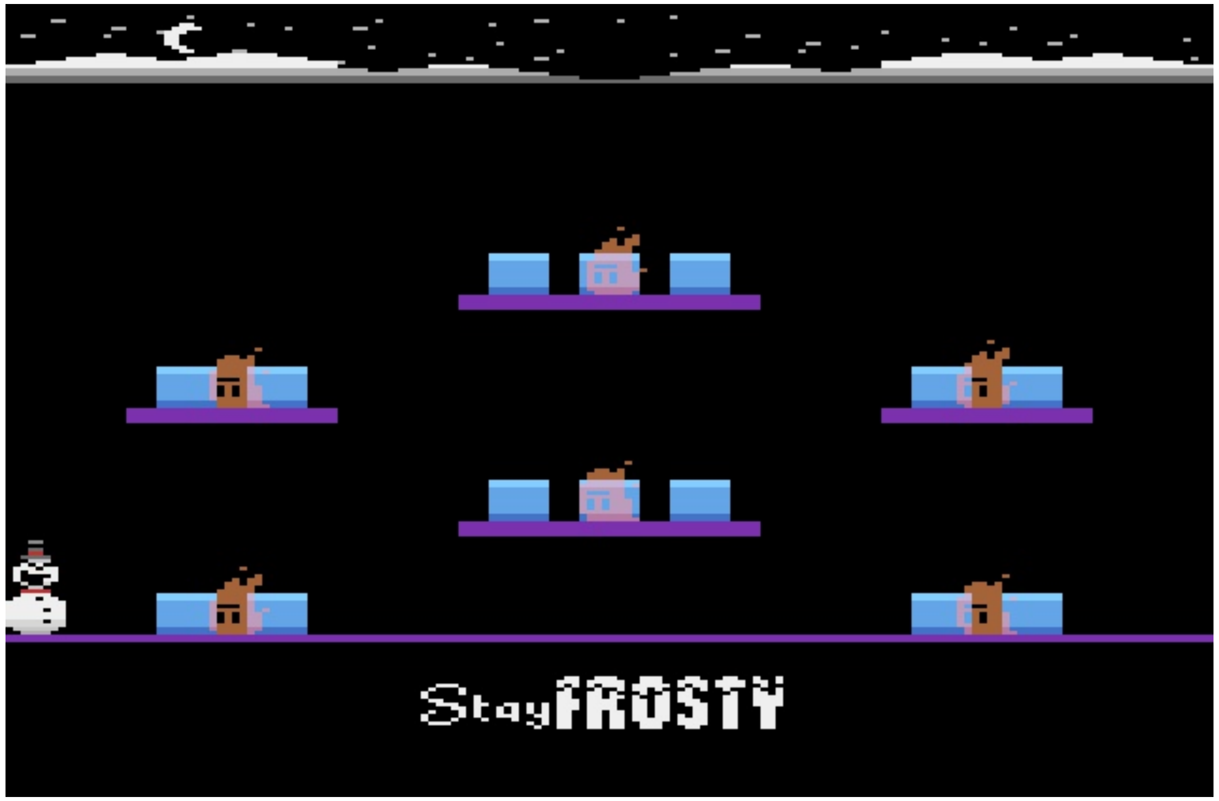


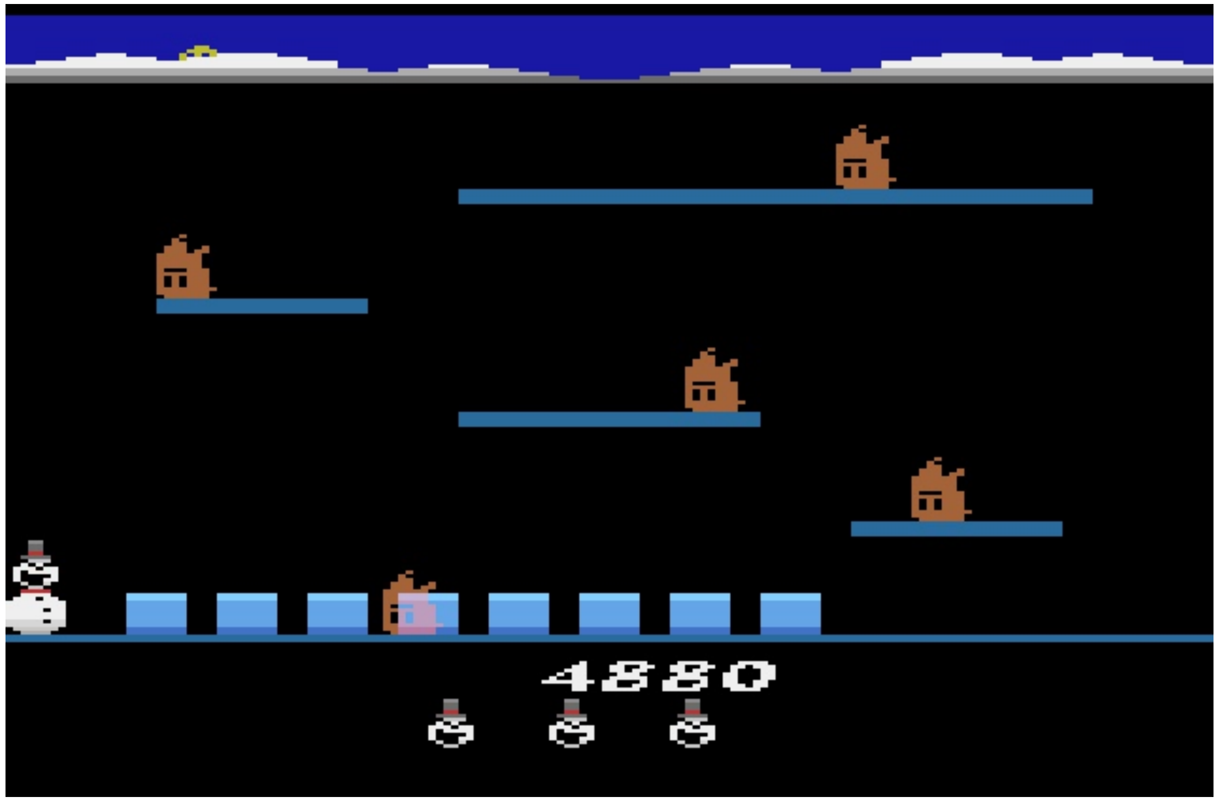


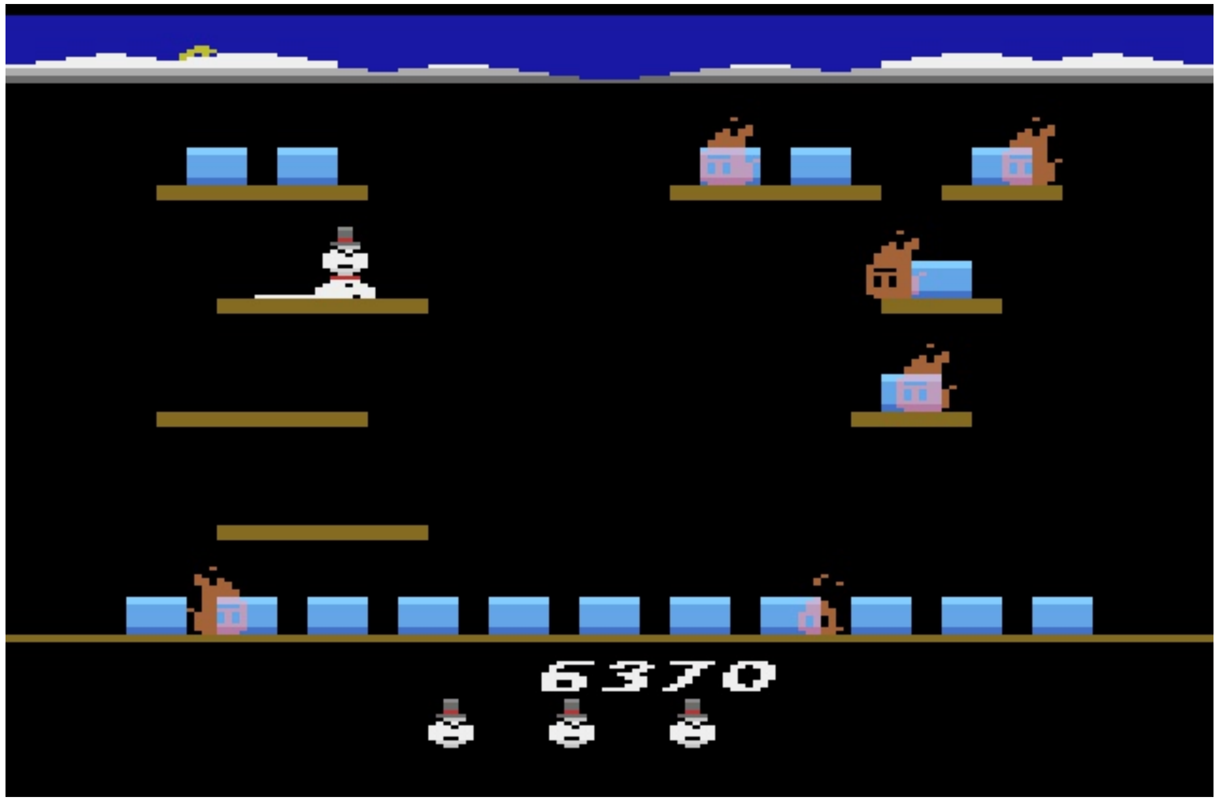
Stay Frosty



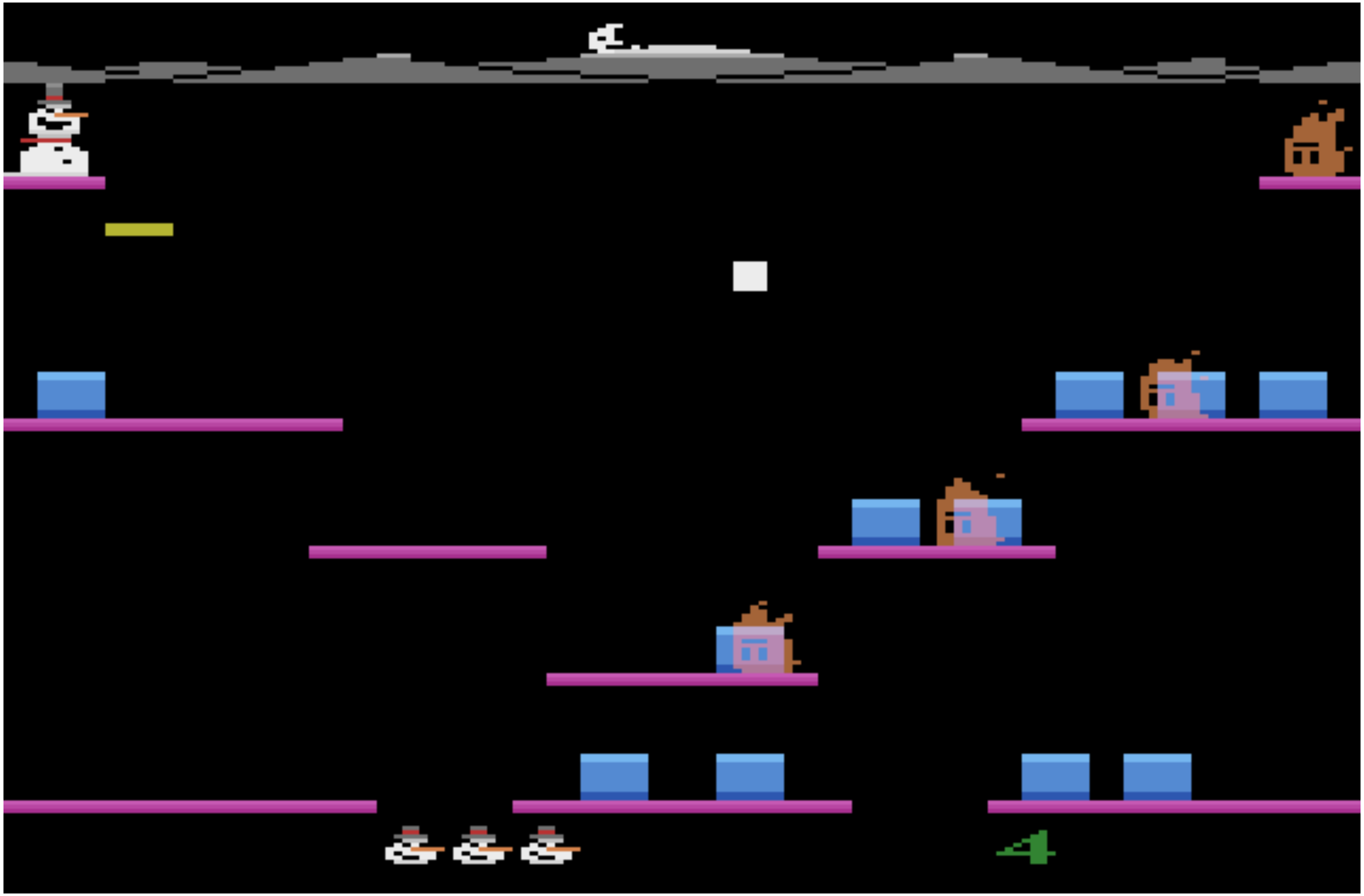

Stella's
Stockings
WELCOME

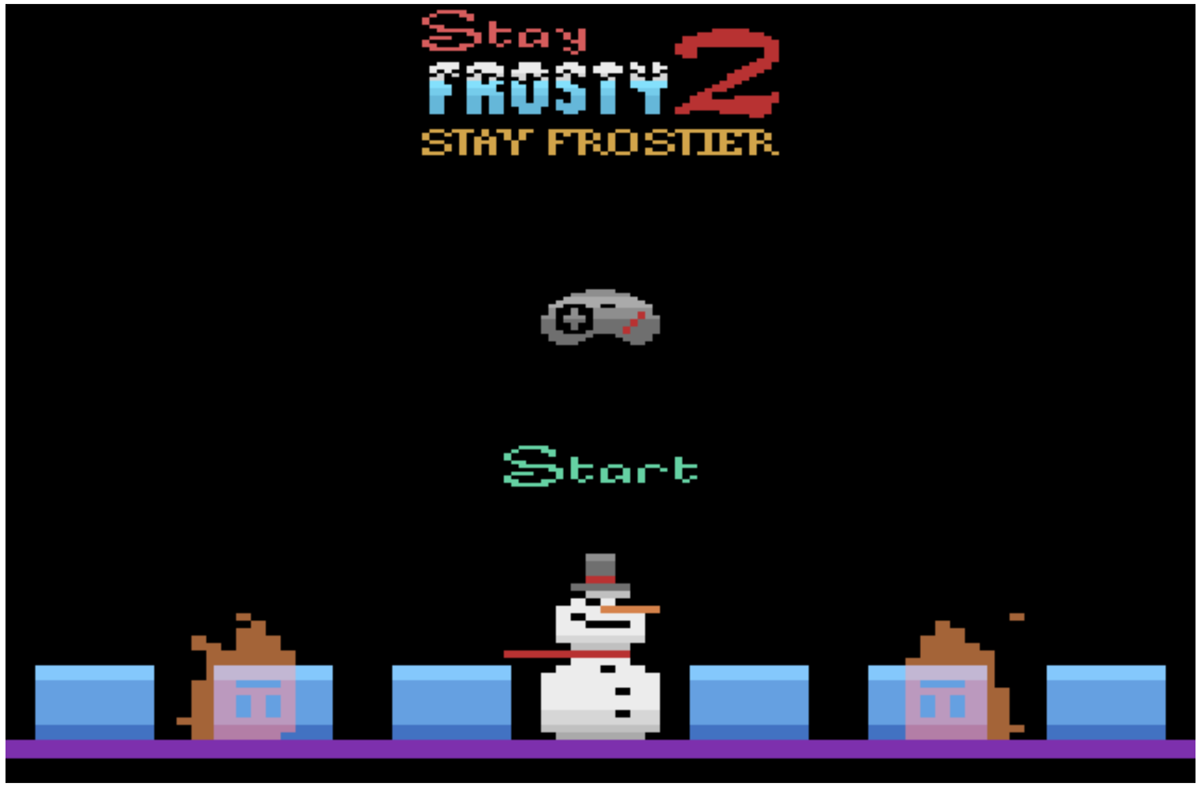

Stay FROSTY

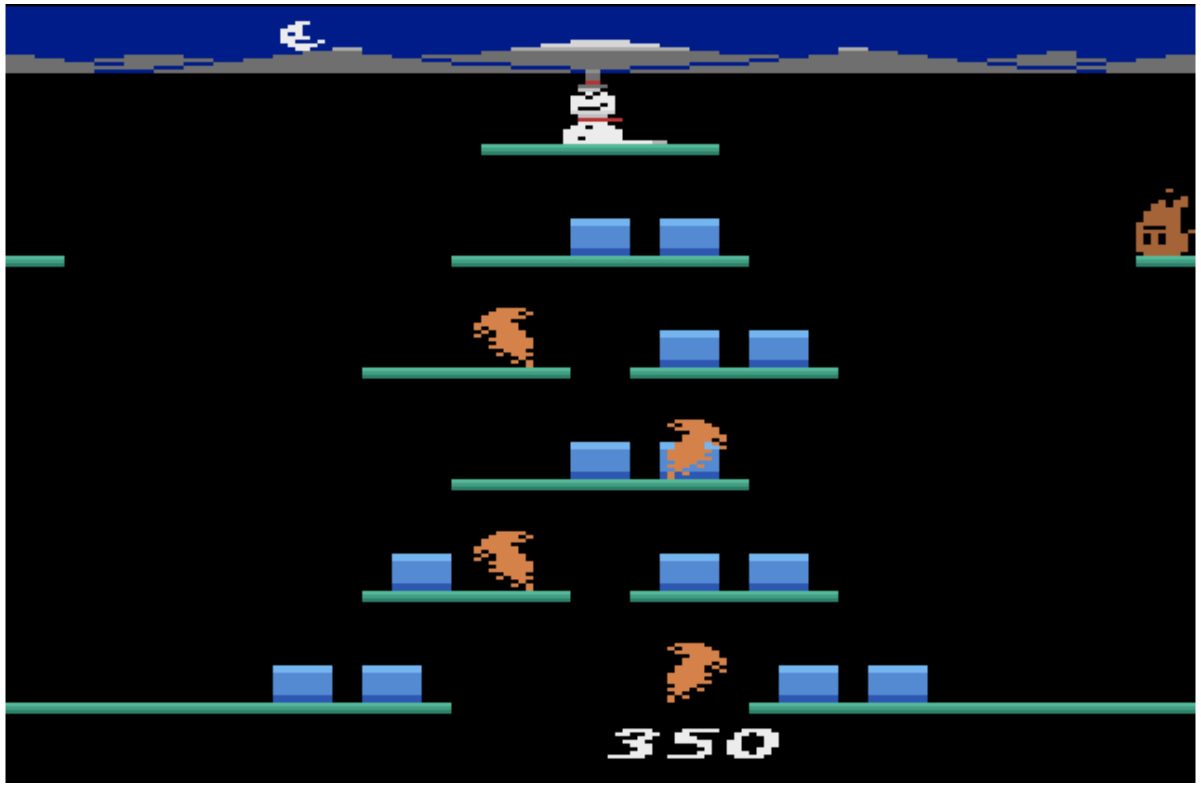

4880

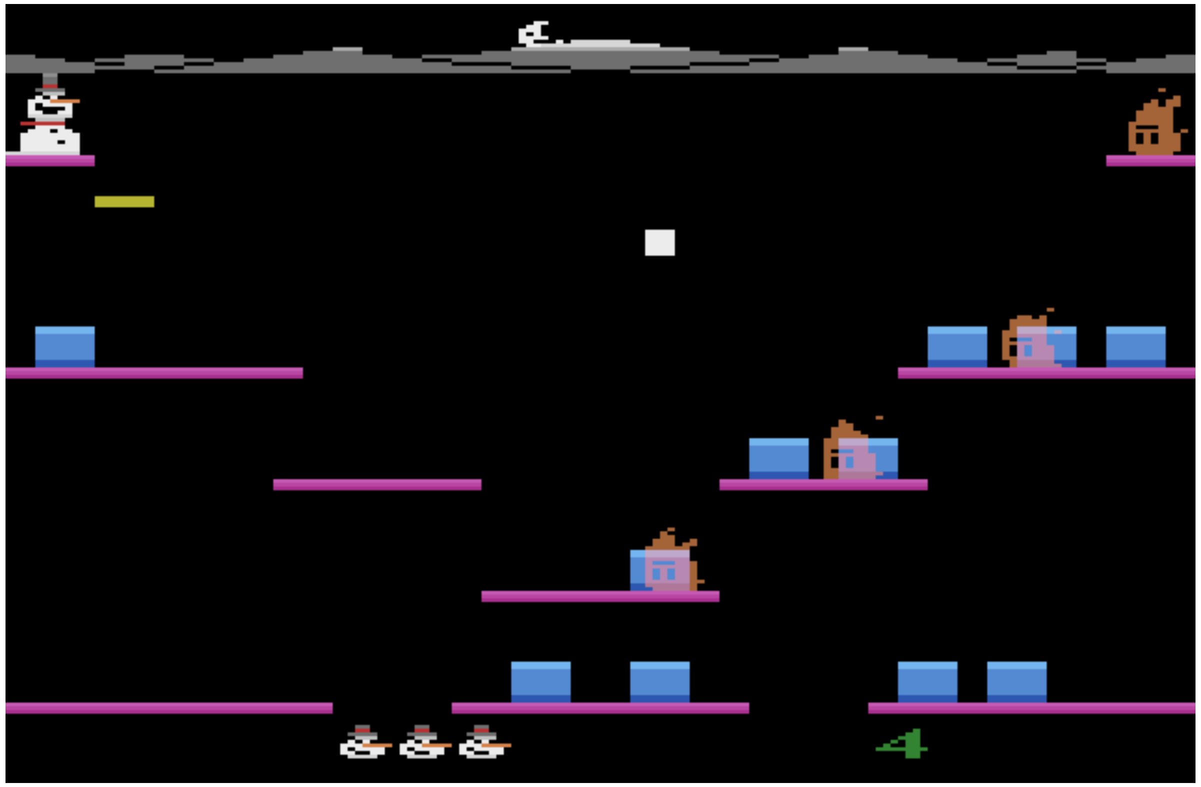

6370

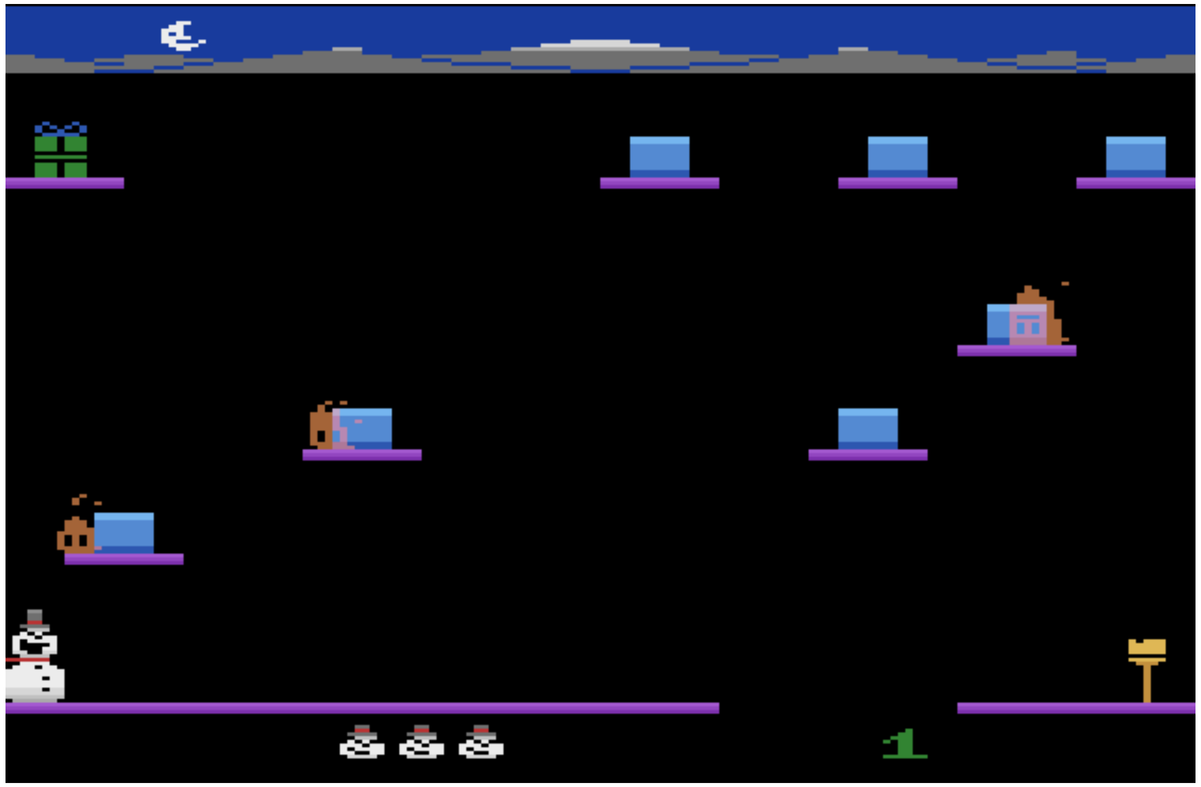
Stay Frosty 2



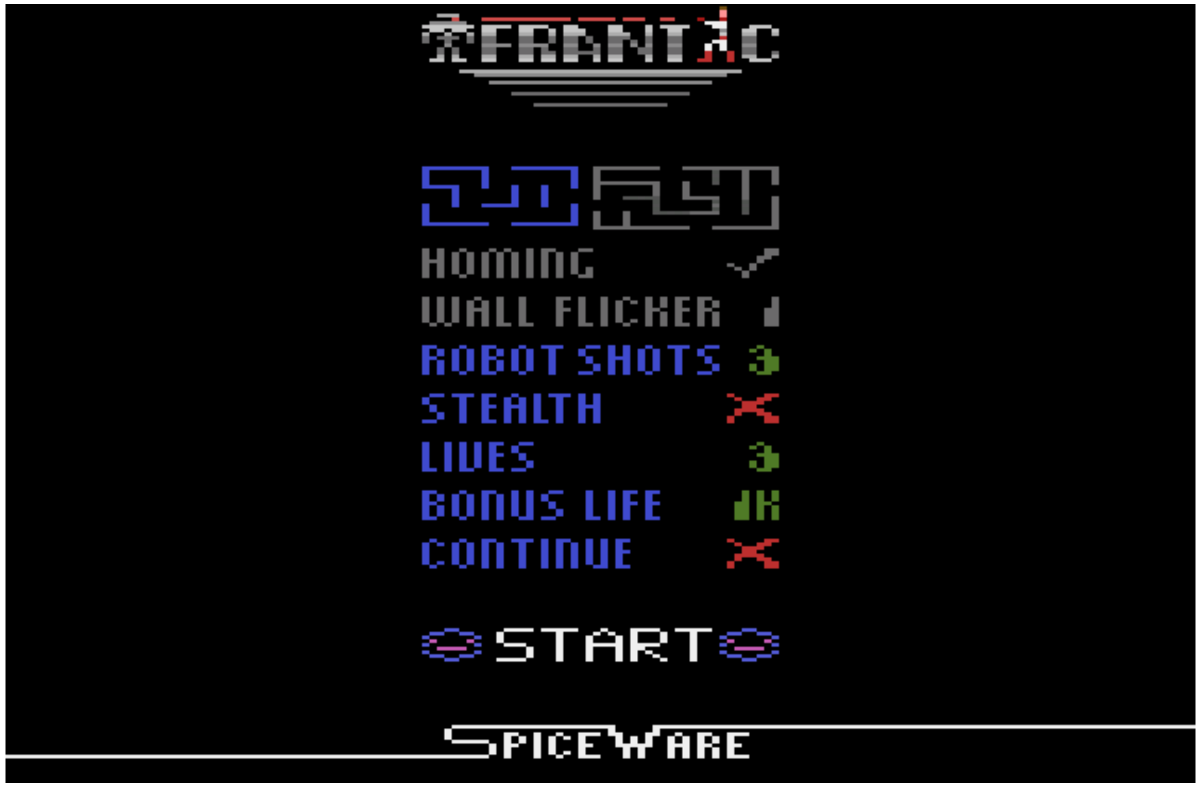

Stay
FROSTY 2
STAY FROSTIER
Start


350


4


1

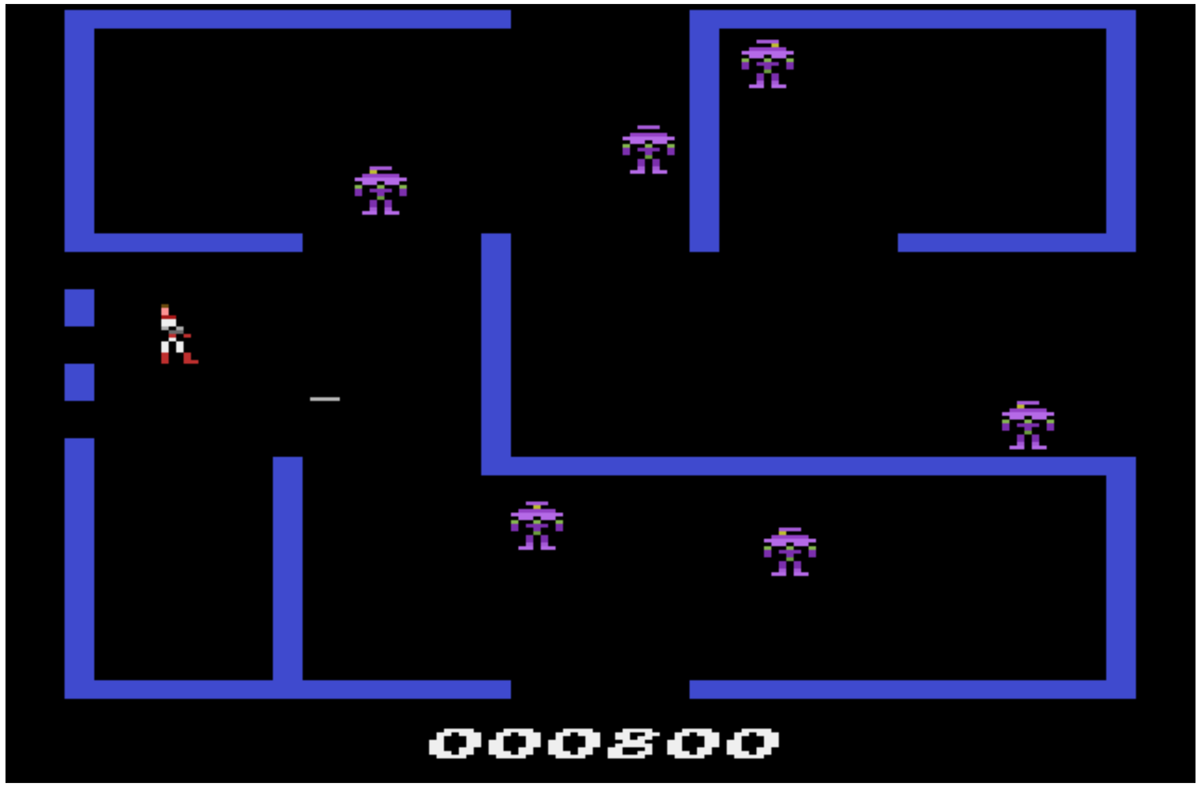
WIP - Frantic

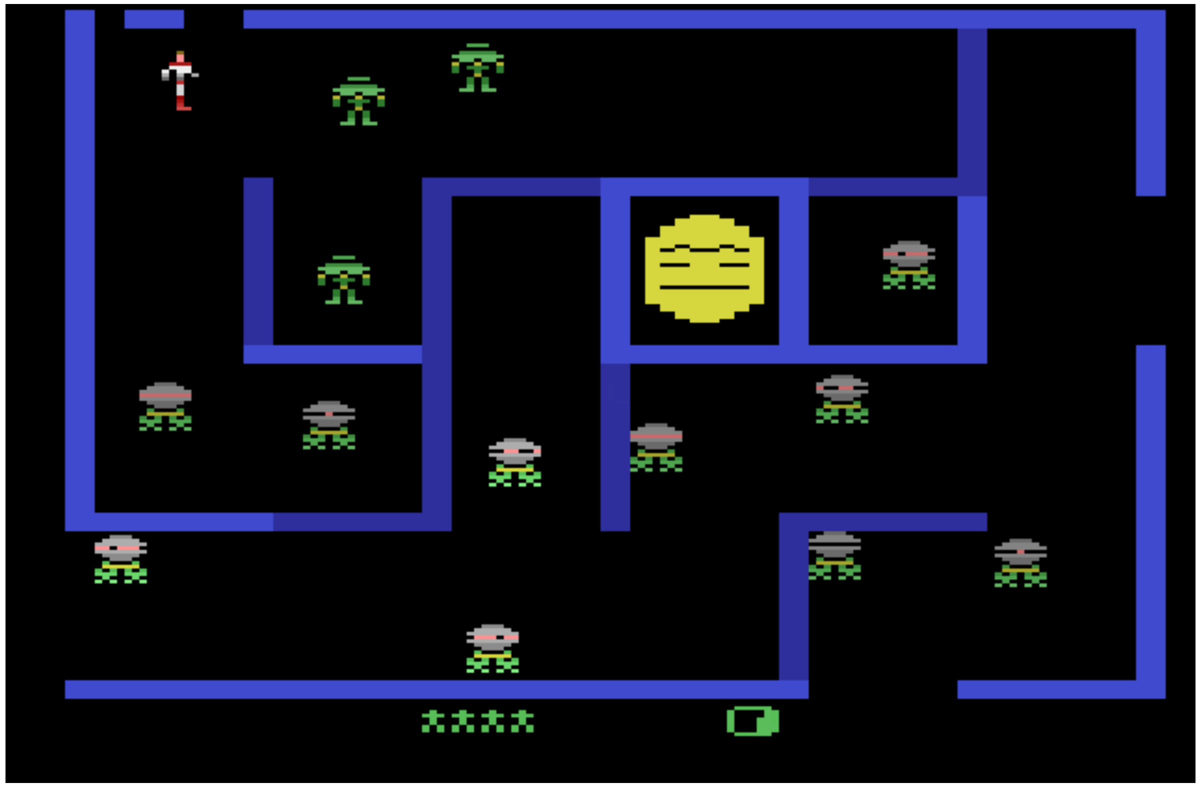

FRANTIC

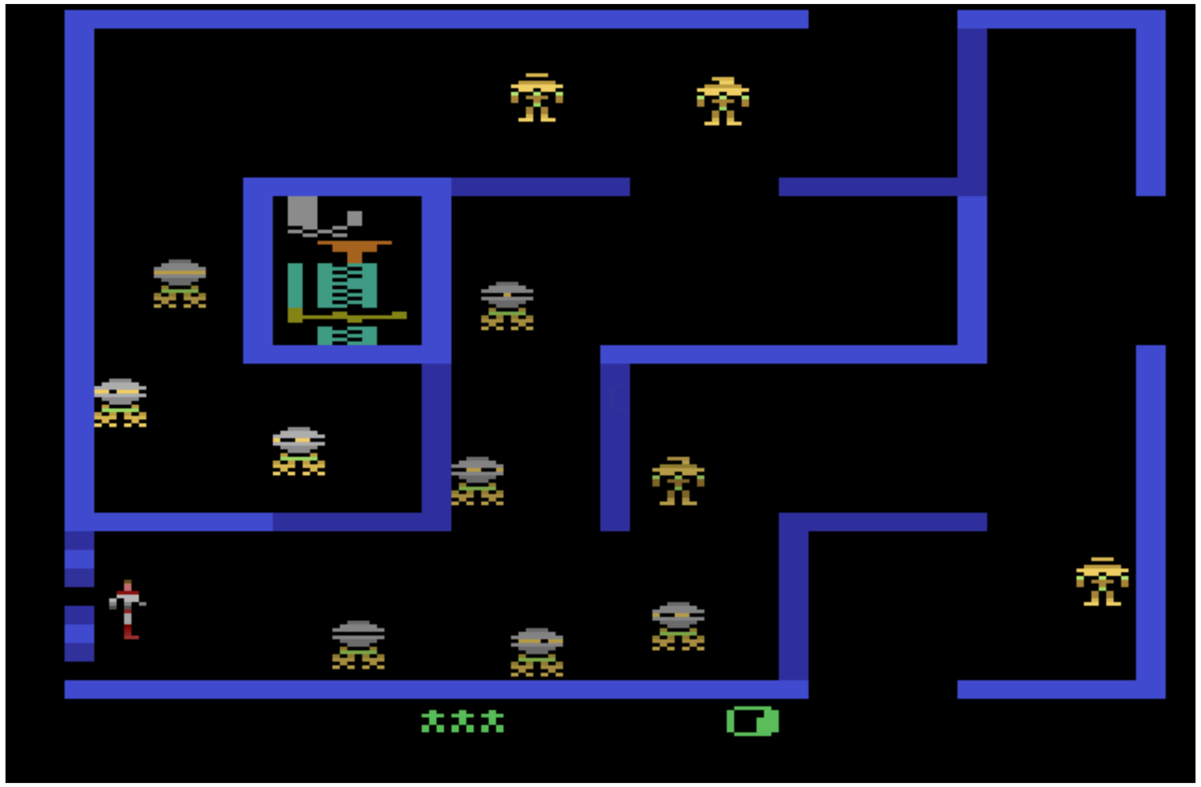
HOMING ✓
WALL FLICKER 1
ROBOT SHOTS 3
STEALTH ✗
LIVES 3
BONUS LIFE 1K
CONTINUE ✗

START

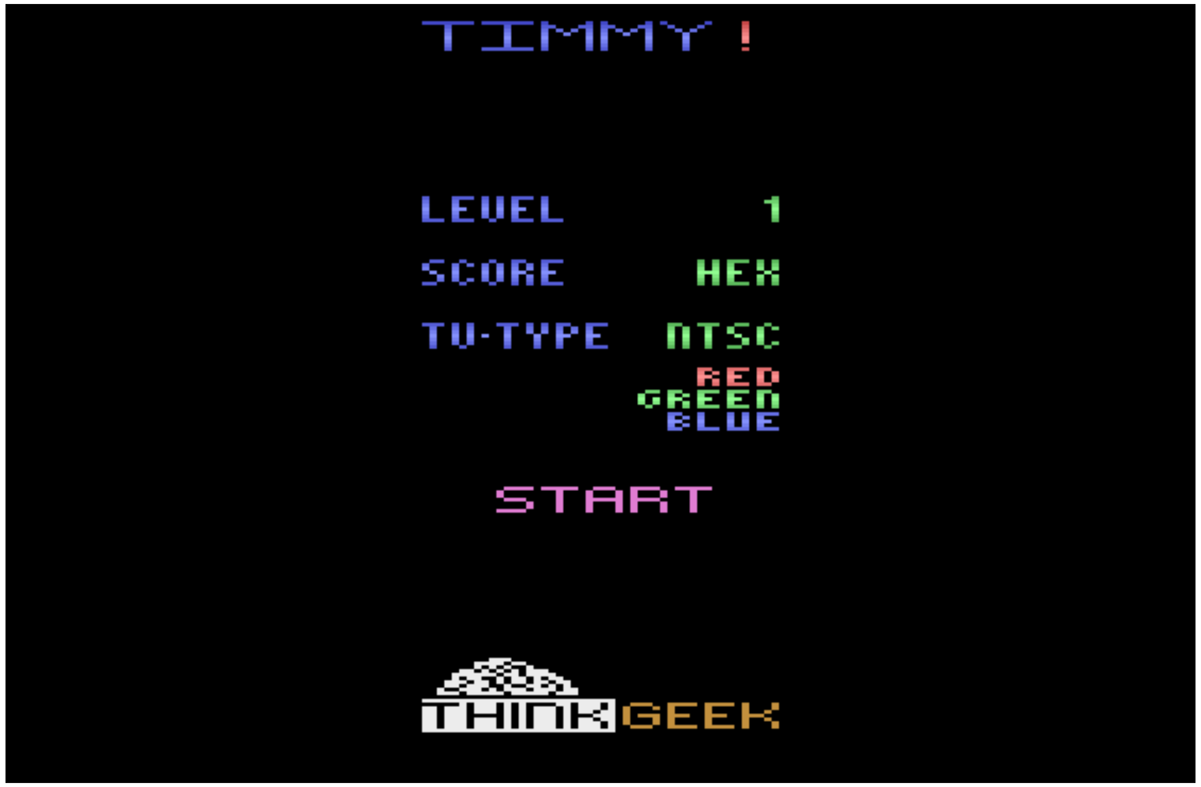
SPICEWARE







WIP - TIMMY!



TIMMY !

LEVEL 1

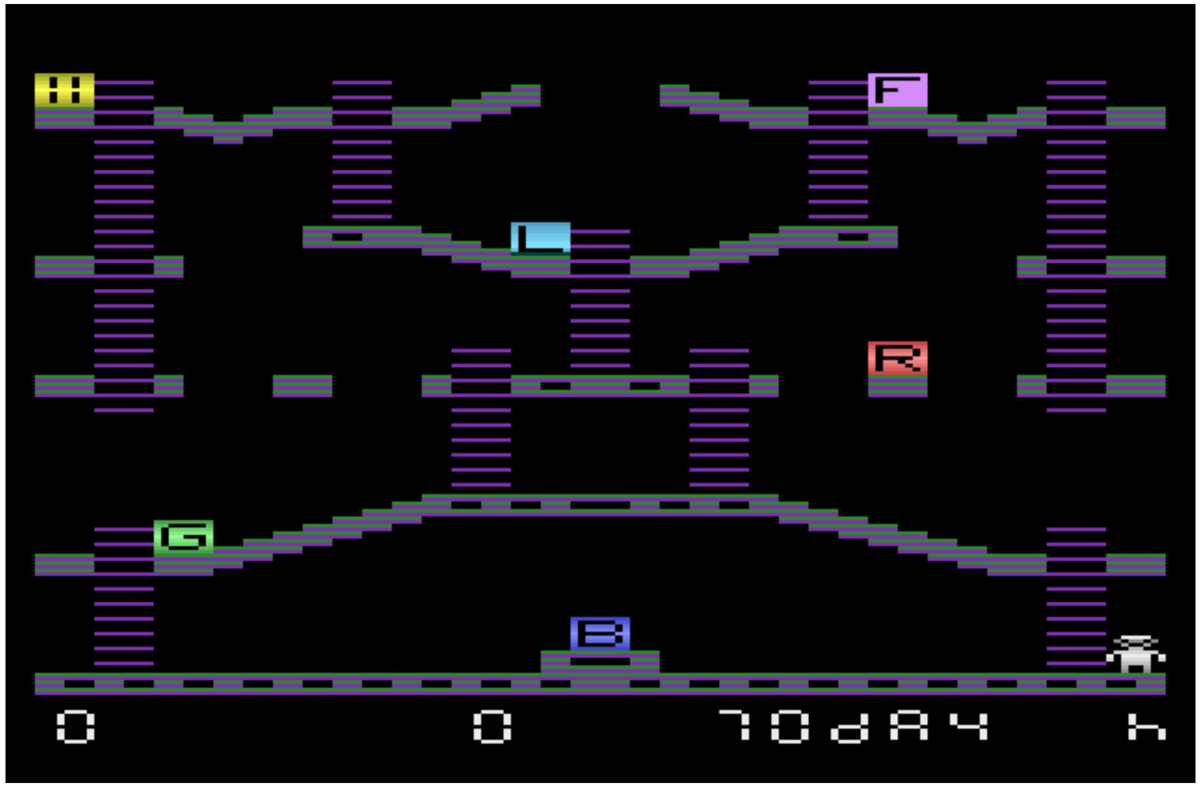
SCORE HEX

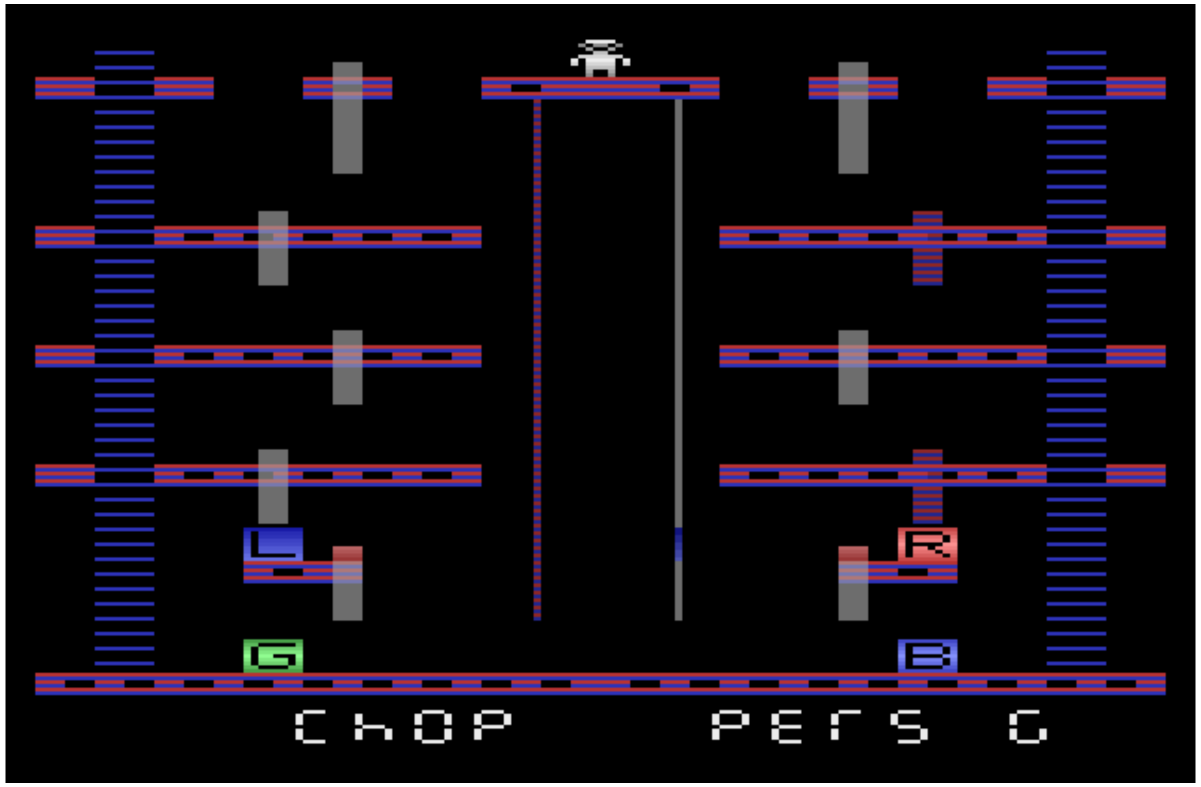
TV-TYPE NTSC

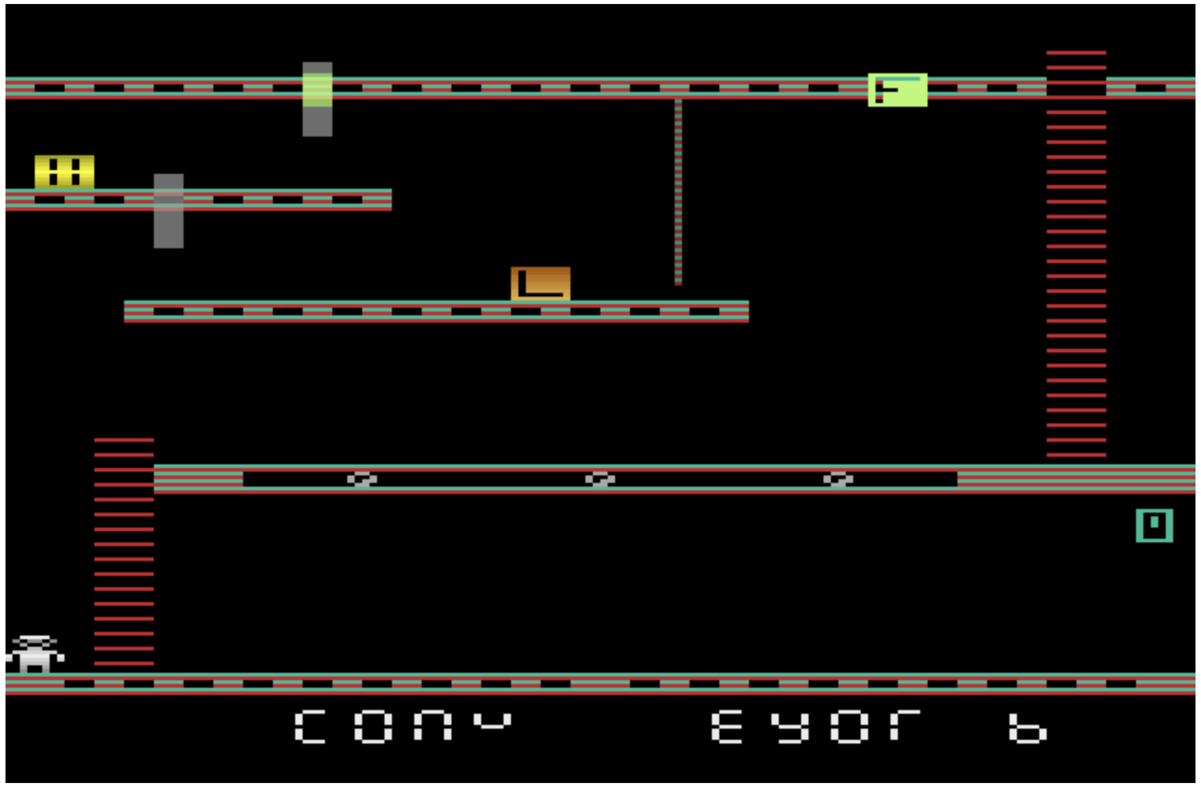
RED
GREEN
BLUE

START

THINKGEEK







Challenges

- 128 bytes of system RAM (1/8 KB)
  PS3 has 256 MB (262,144 KB)
- no video RAM
  PS3 has 256 MB
- 4K cartridge space
- 1 MHz CPU
  only 27% is available for game logic

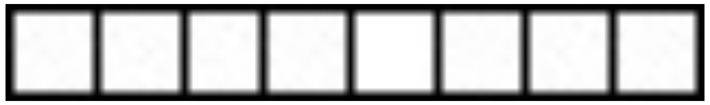
128 bytes of RAM

- 1 KB cost $66 in 1975 when work began on Stella (code name for the Atari)
- 128 bytes = 1/8 KB, about $8.25

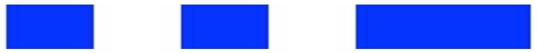
No video RAM

- TIA - Television Interface Adaptor is scan line based
- 2 players (sprites)
- 2 missiles
- 1 ball
- Low resolution playfield

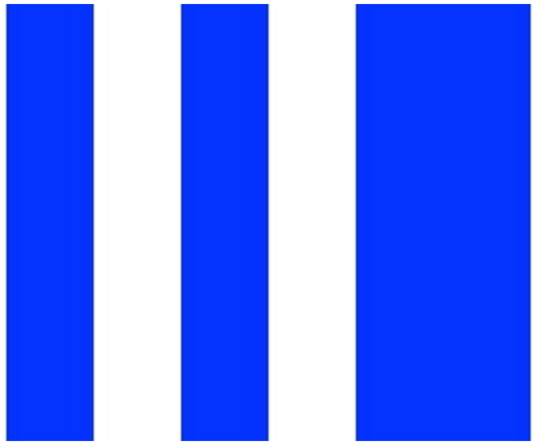
2 Players



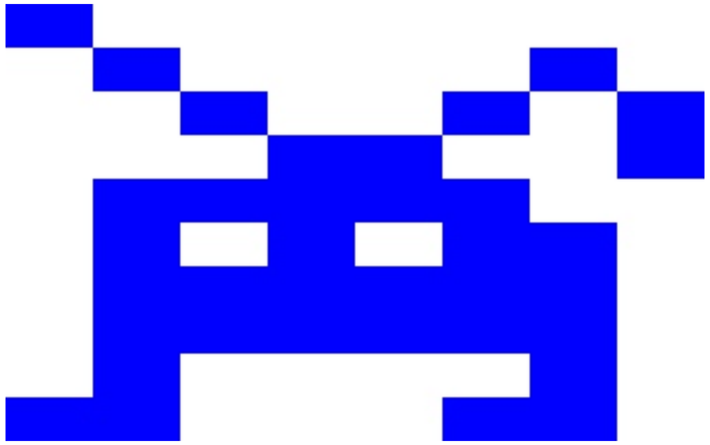
8 x 1 image



Sample pattern



If not changed, pattern repeats down the screen



Shapes are created by changing pattern on each scan line

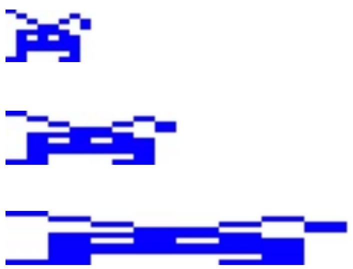
Player Features

Three sizes:

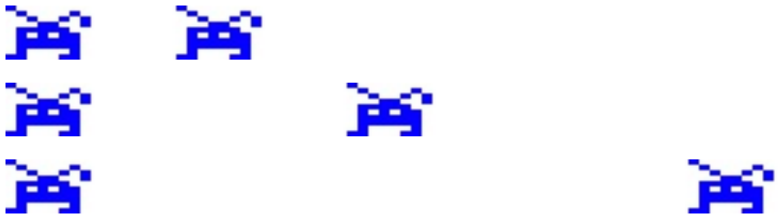
1x

2x

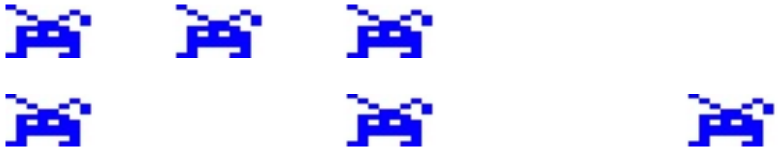
4x



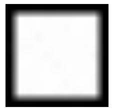
Three 1x duplicates



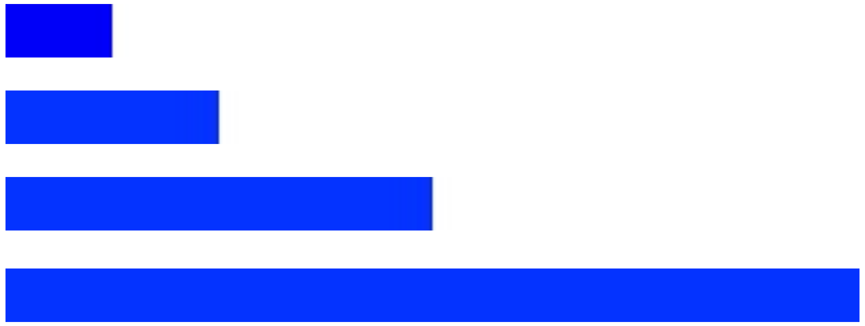
Two 1x triplicates



2 Missiles, 1 Ball



1 x 1 image



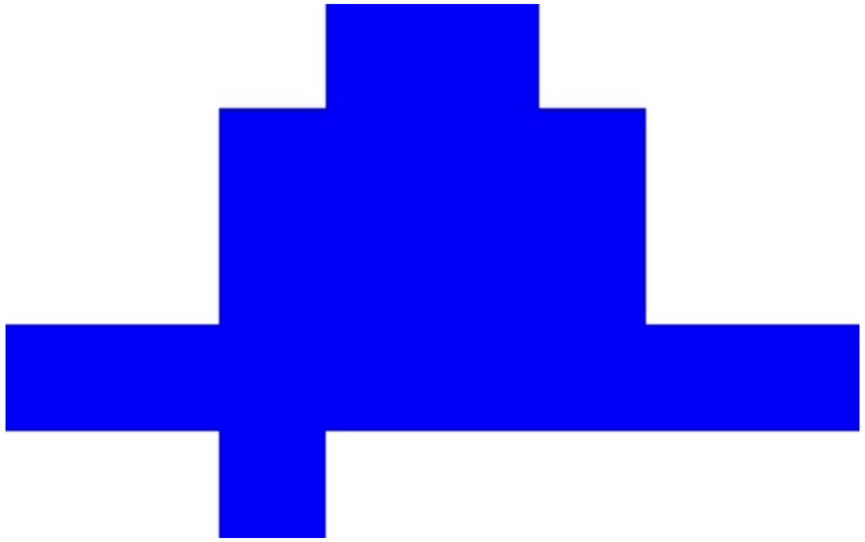
Four sizes:

1x

2x

4x

8x

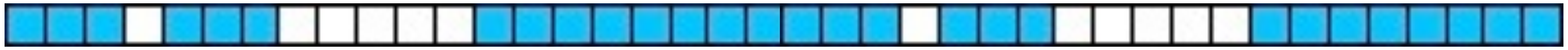


Can be used to
create objects like
this bell

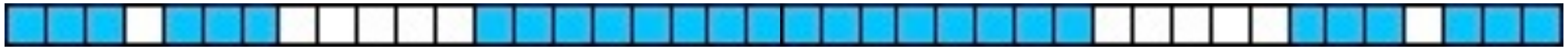
Playfield

20 x 1 image

Playfield is repeated



or reflected



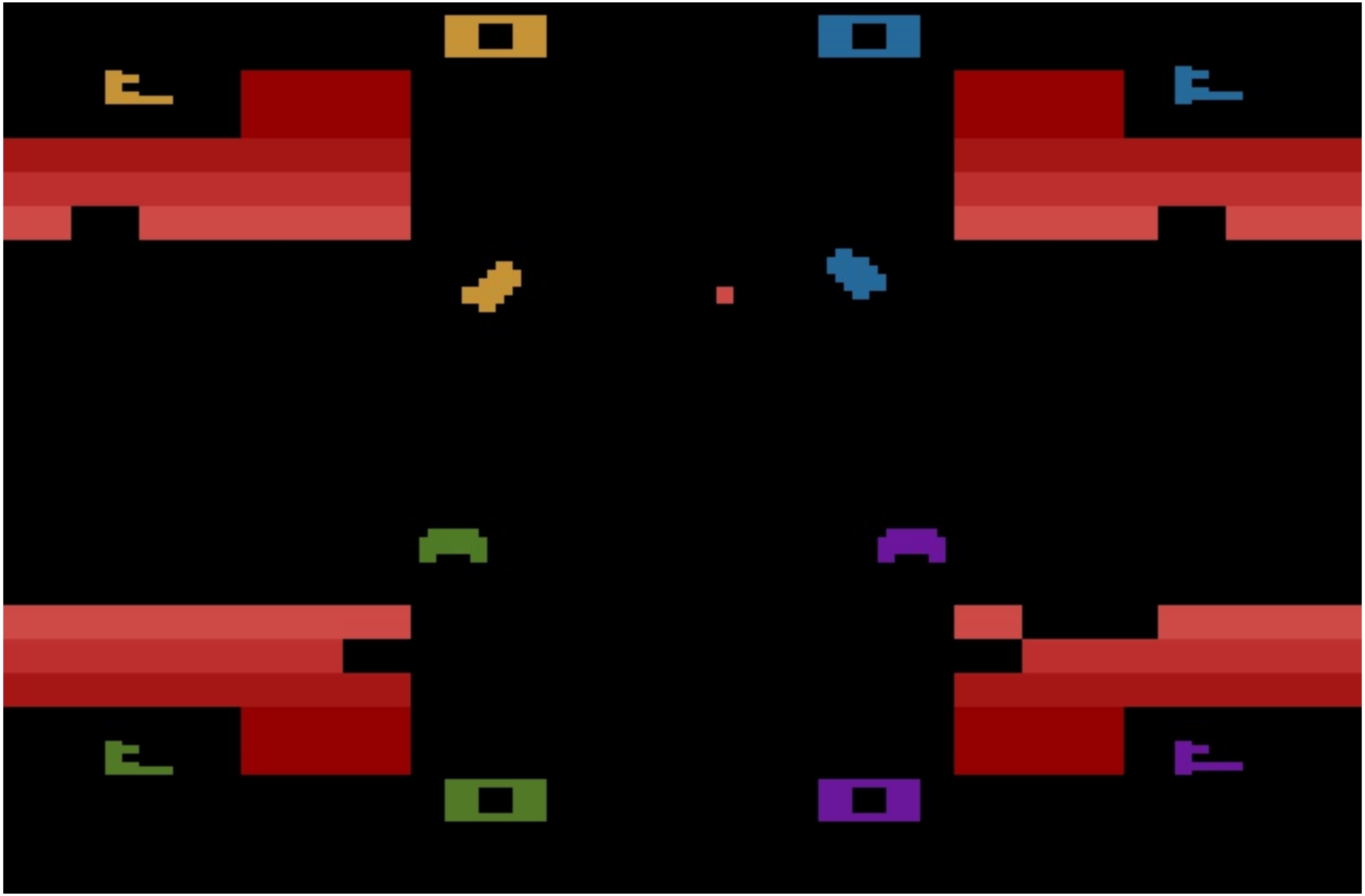
to fill width of screen

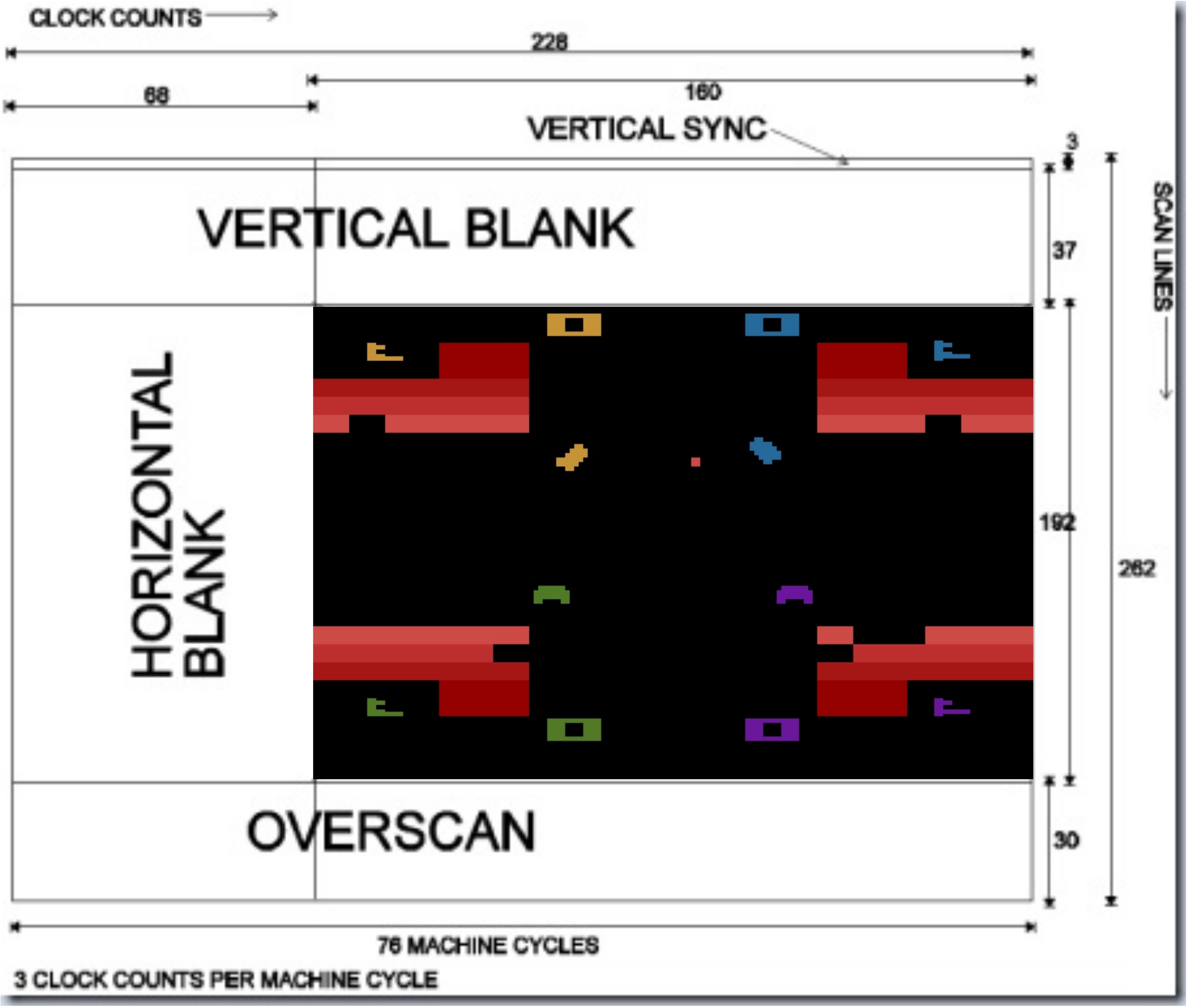
4K Cartridge

- original games were 2K
- 4K was believed to be large enough to last until the 2600's replacement hit the market
- ROM only, no Read/Write line for controlling access to RAM

27% of 1 MHz

- CPU must update TIA (scan line video chip) in real time
- Portion of program that drives TIA is known as the Kernel
- CPU must also trigger sync signal for TV




CLOCK COUNTS
228
68
160
VERTICAL SYNC
3
VERTICAL BLANK
37
HORIZONTAL BLANK
192
262
SCAN LINES
OVERSCAN
30
76 MACHINE CYCLES
3 CLOCK COUNTS PER MACHINE CYCLE

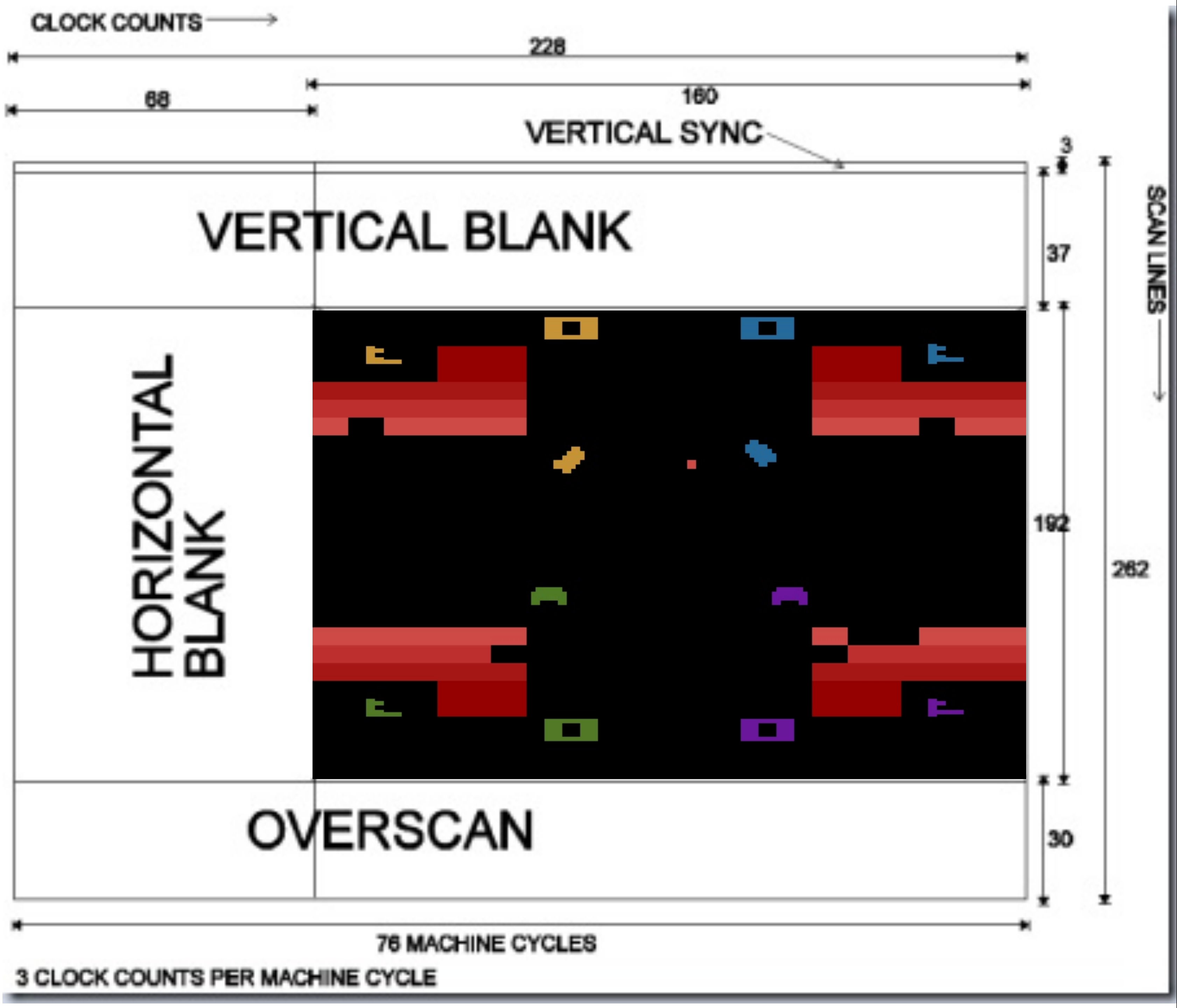
Program flow

Sync signal

Game logic

Kernel
(draw display)

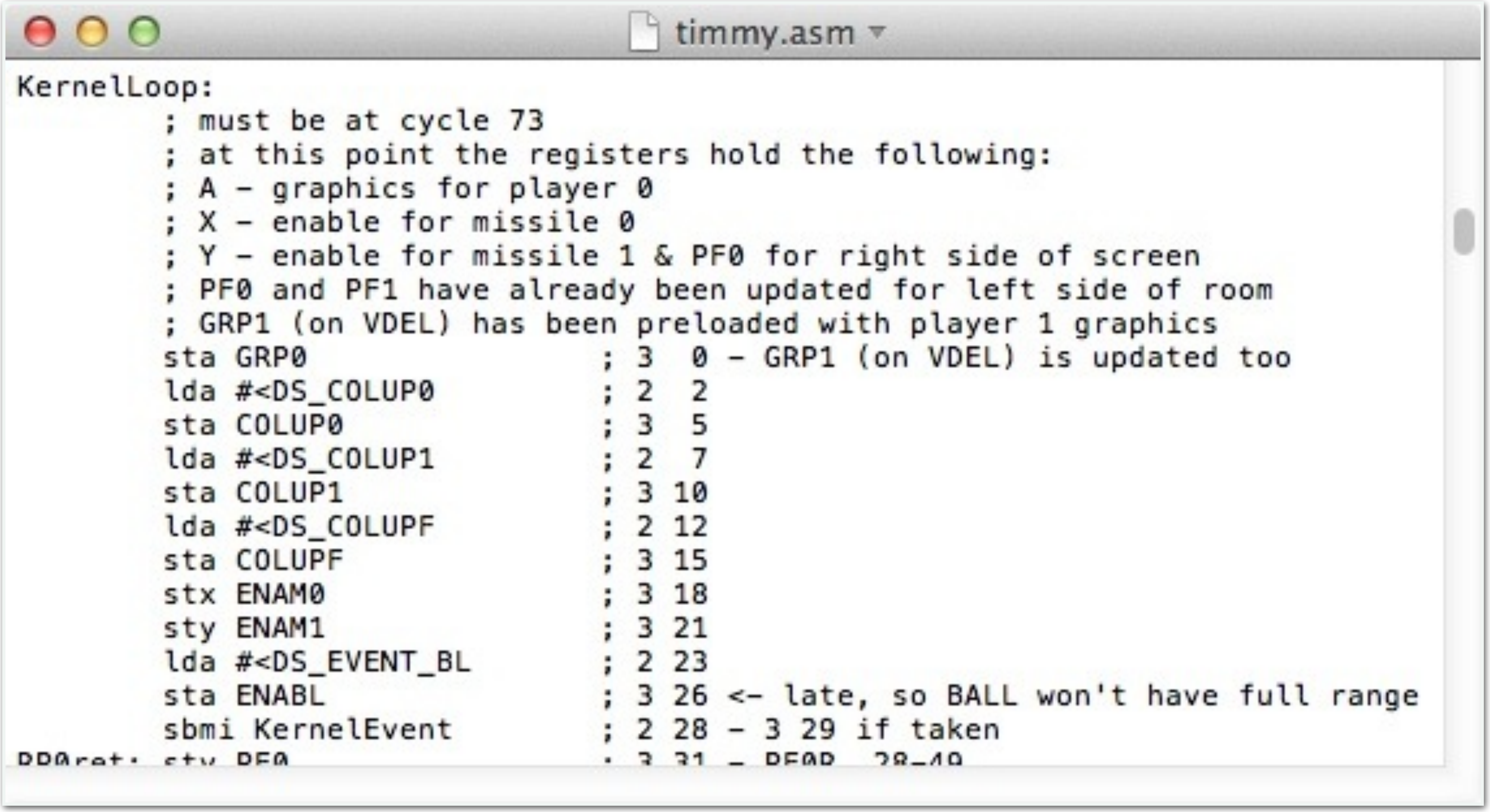
Game logic



What do you need?

- Editor
- Dasm
- Stella
- Hardware

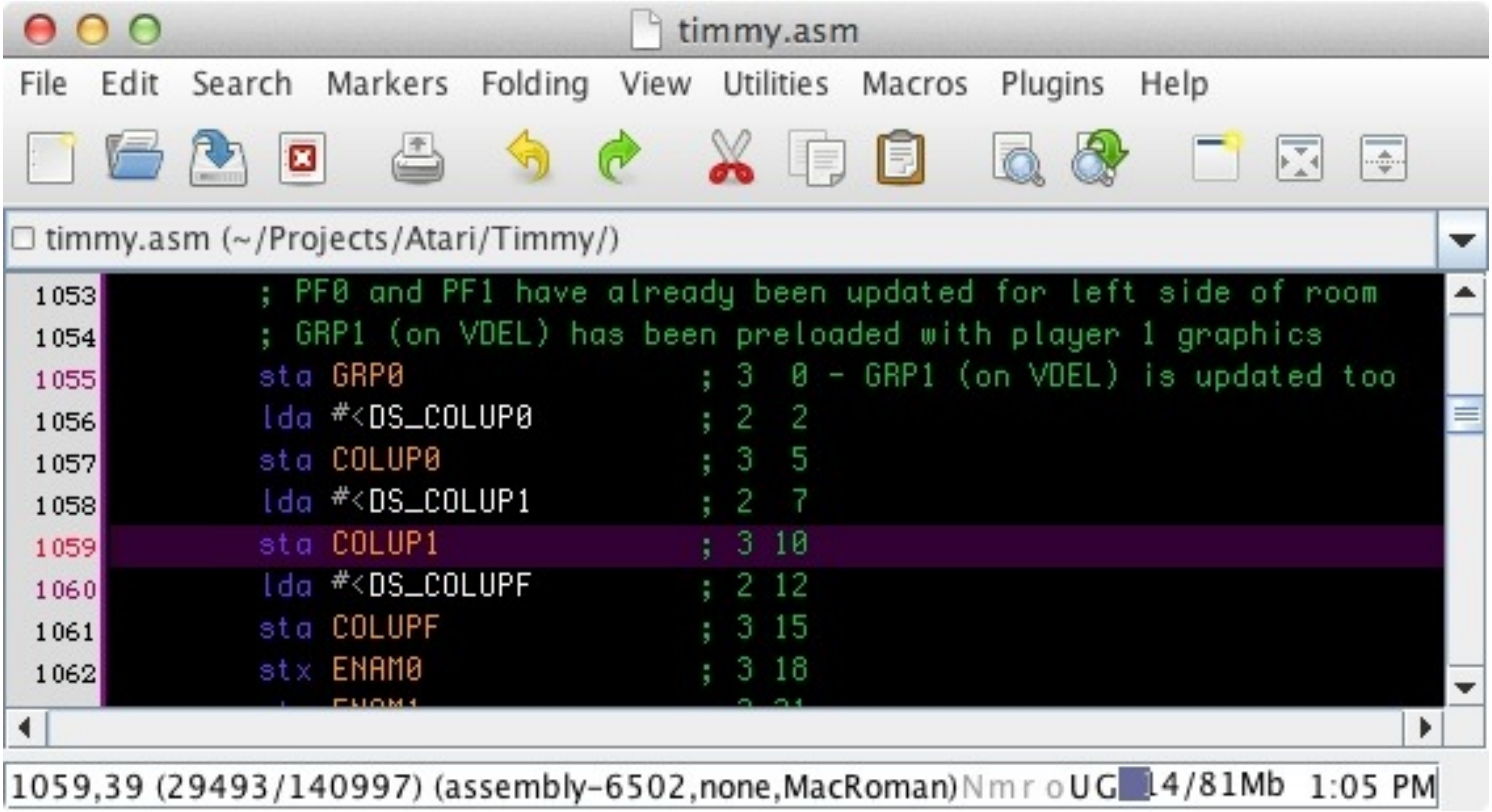
Editor



```
KernelLoop:
        ; must be at cycle 73
        ; at this point the registers hold the following:
        ; A - graphics for player 0
        ; X - enable for missile 0
        ; Y - enable for missile 1 & PF0 for right side of screen
        ; PF0 and PF1 have already been updated for left side of room
        ; GRP1 (on VDEL) has been preloaded with player 1 graphics
        sta GRP0                ; 3  0 - GRP1 (on VDEL) is updated too
        lda #<DS_COLUP0         ; 2  2
        sta COLUP0              ; 3  5
        lda #<DS_COLUP1         ; 2  7
        sta COLUP1              ; 3 10
        lda #<DS_COLUPF         ; 2 12
        sta COLUPF              ; 3 15
        stx ENAM0               ; 3 18
        sty ENAM1               ; 3 21
        lda #<DS_EVENT_BL       ; 2 23
        sta ENABL               ; 3 26 <- late, so BALL won't have full range
        sbmi KernelEvent        ; 2 28 - 3 29 if taken
PP0ret: sty PF0                 ; 3 31 - PF0R  28-49
```

Notepad or TextEdit will do

A programmer's editor like jEdit is nicer
http://www.jedit.org



```
        ; PF0 and PF1 have already been updated for left side of room
        ; GRP1 (on VDEL) has been preloaded with player 1 graphics
        sta GRP0                ; 3  0 - GRP1 (on VDEL) is updated too
        lda #<DS_COLUP0         ; 2  2
        sta COLUP0              ; 3  5
        lda #<DS_COLUP1         ; 2  7
        sta COLUP1              ; 3 10
        lda #<DS_COLUPF         ; 2 12
        sta COLUPF              ; 3 15
        stx ENAM0               ; 3 18
```

```
1059,39 (29493/140997) (assembly-6502,none,MacRoman)Nmr oUG 14/81Mb  1:05 PM
```

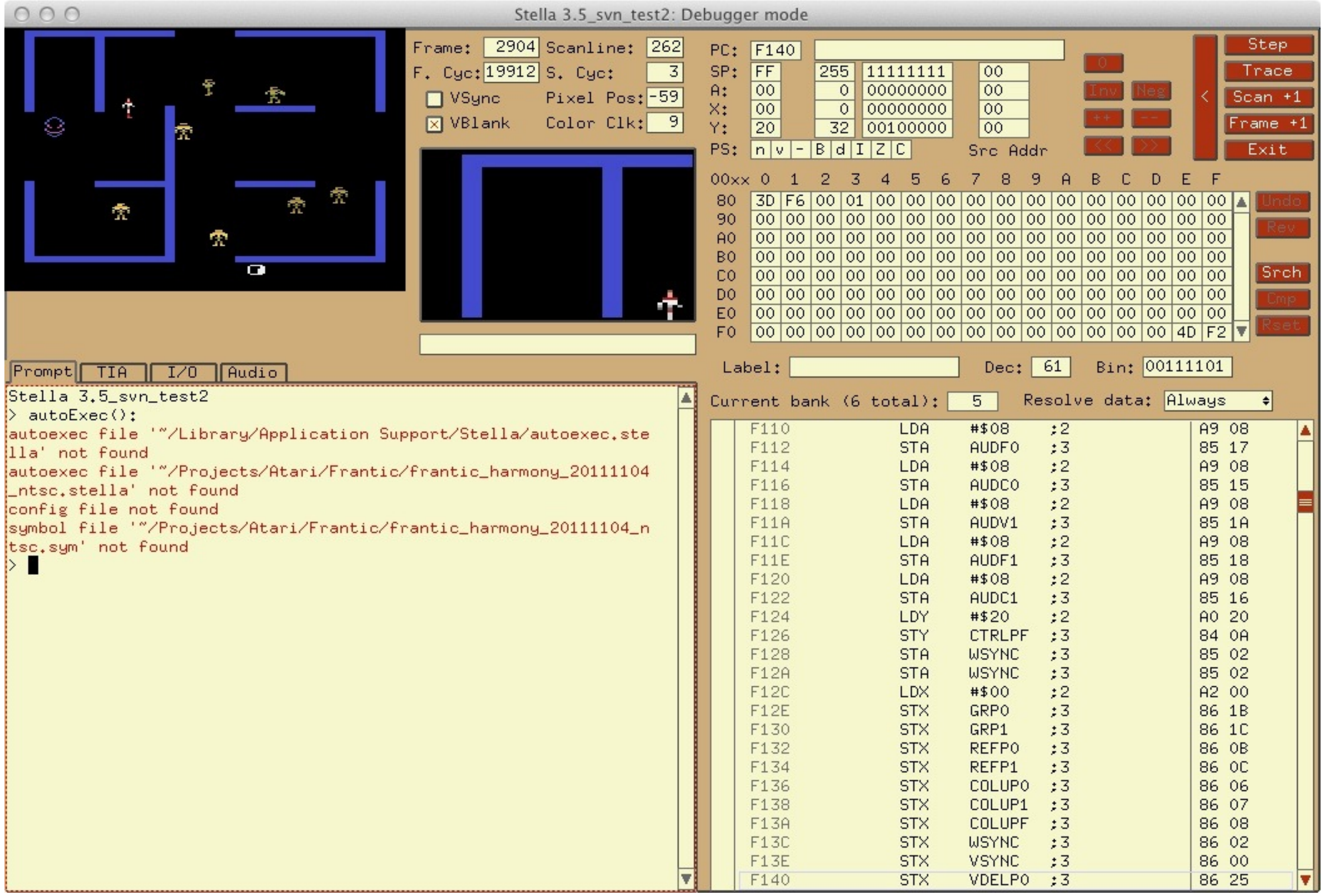
Dasm

Assembler that converts human readable code into machine readable code

http://dasm-dillon.sourceforge.net

Stella

- Turns your computer into an Atari
- Integrated debugger makes coding easier



```
Frame:   2904  Scanline:   262
F. Cyc: 19912  S. Cyc:       3
□ VSync        Pixel Pos: -59
☒ VBlank       Color Clk:   9
```

```
PC:  F140
SP:  FF     255  11111111   00
A:   00       0  00000000   00
X:   00       0  00000000   00
Y:   20      32  00100000   00
PS:  n v - B d I Z C        Src Addr
```

```
00xx  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 80  3D F6 00 01 00 00 00 00 00 00 00 00 00 00 00 00
 90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 4D F2
```

```
Label:                 Dec:  61   Bin: 00111101
Current bank (6 total):  5   Resolve data: Always
```

```
Stella 3.5_svn_test2
> autoExec():
autoexec file '~/Library/Application Support/Stella/autoexec.ste
lla' not found
autoexec file '~/Projects/Atari/Frantic/frantic_harmony_20111104
_ntsc.stella' not found
config file not found
symbol file '~/Projects/Atari/Frantic/frantic_harmony_20111104_n
tsc.sym' not found
> 
```

```
F110        LDA    #$08    ;2      A9 08
F112        STA    AUDF0   ;3      85 17
F114        LDA    #$08    ;2      A9 08
F116        STA    AUDC0   ;3      85 15
F118        LDA    #$08    ;2      A9 08
F11A        STA    AUDV1   ;3      85 1A
F11C        LDA    #$08    ;2      A9 08
F11E        STA    AUDF1   ;3      85 18
F120        LDA    #$08    ;2      A9 08
F122        STA    AUDC1   ;3      85 16
F124        LDY    #$20    ;2      A0 20
F126        STY    CTRLPF  ;3      84 0A
F128        STA    WSYNC   ;3      85 02
F12A        STA    WSYNC   ;3      85 02
F12C        LDX    #$00    ;2      A2 00
F12E        STX    GRP0    ;3      86 1B
F130        STX    GRP1    ;3      86 1C
F132        STX    REFP0   ;3      86 0B
F134        STX    REFP1   ;3      86 0C
F136        STX    COLUP0  ;3      86 06
F138        STX    COLUP1  ;3      86 07
F13A        STX    COLUPF  ;3      86 08
F13C        STX    WSYNC   ;3      86 02
F13E        STX    VSYNC   ;3      86 00
F140        STX    VDELP0  ;3      86 25
```

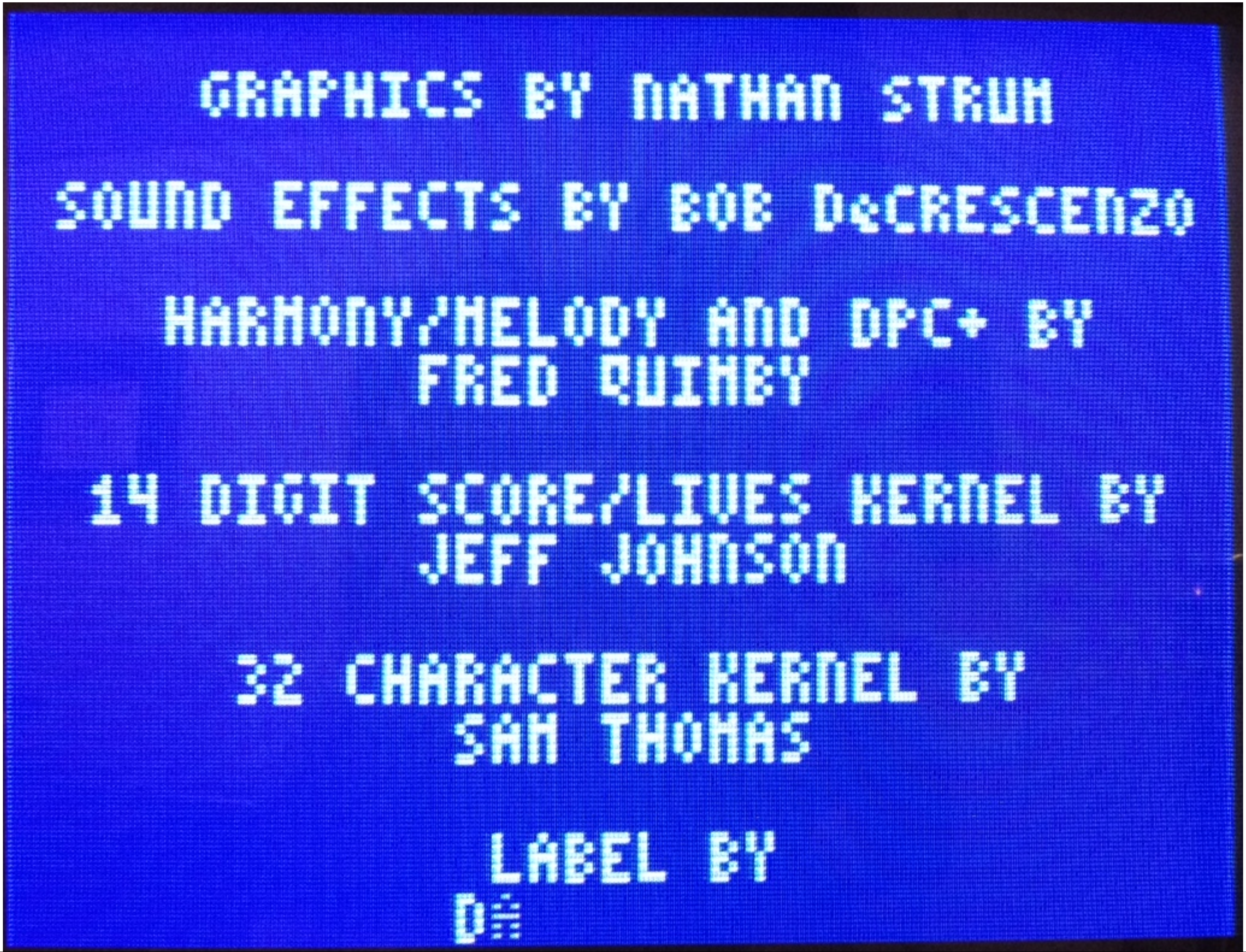
Hardware

- Atari 2600
- Supercharger
- Krokodile cart
- Harmony

Atari 2600

While Stella is great, it's not 100% accurate so you need to test your code on the real thing.

32 character text on Atari



GRAPHICS BY NATHAN STRUH

SOUND EFFECTS BY BOB DeCRESCENZO

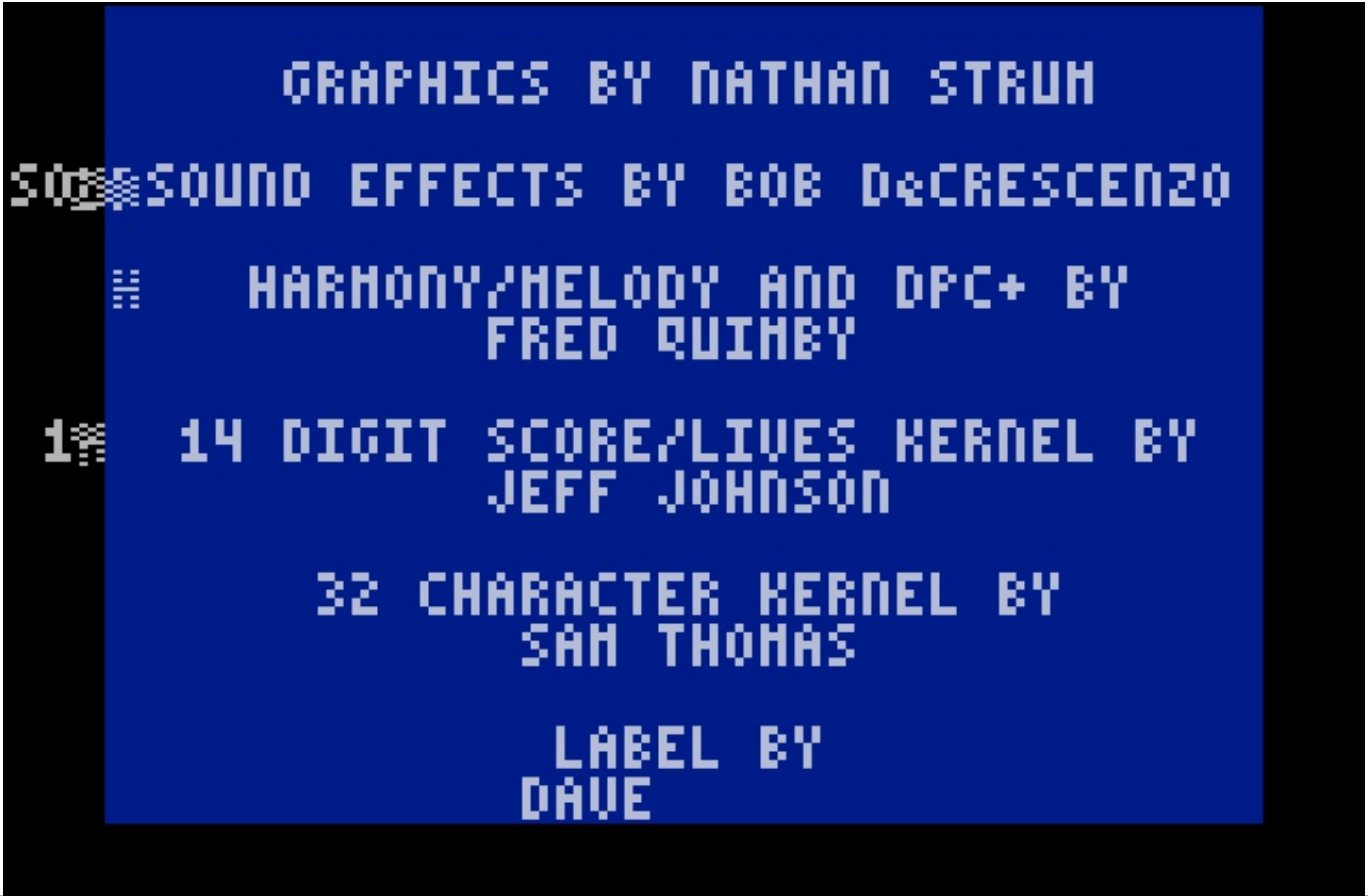
HARMONY/MELODY AND DPC+ BY
FRED QUIMBY

14 DIGIT SCORE/LIVES KERNEL BY
JEFF JOHNSON

32 CHARACTER KERNEL BY
SAM THOMAS

LABEL BY
D

32 character text on Stella



GRAPHICS BY NATHAN STRUM

SOUND EFFECTS BY BOB DeCRESCENZO

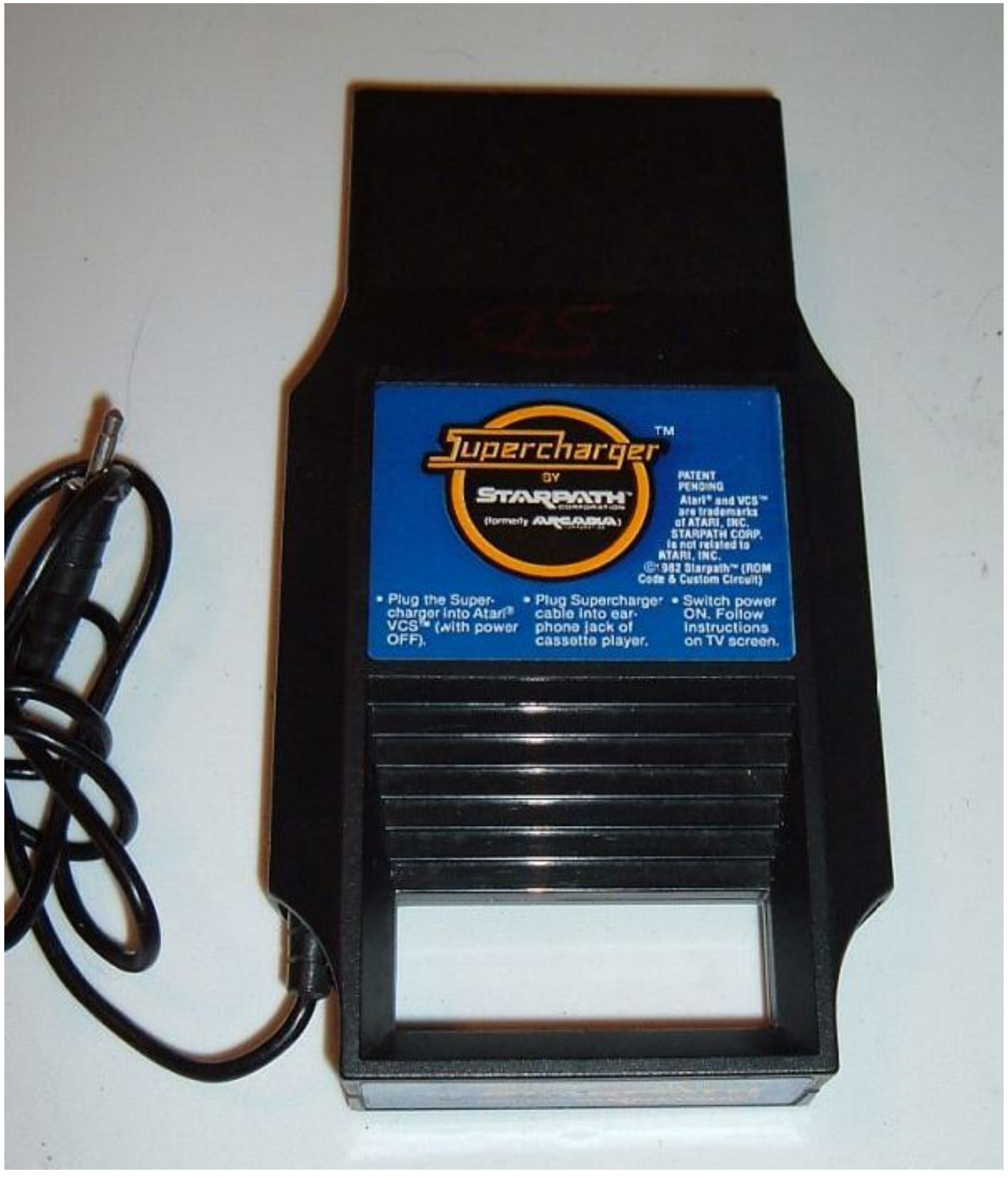
HARMONY/MELODY AND DPC+ BY
FRED QUIMBY

14 DIGIT SCORE/LIVES KERNEL BY
JEFF JOHNSON

32 CHARACTER KERNEL BY
SAM THOMAS

LABEL BY
DAVE

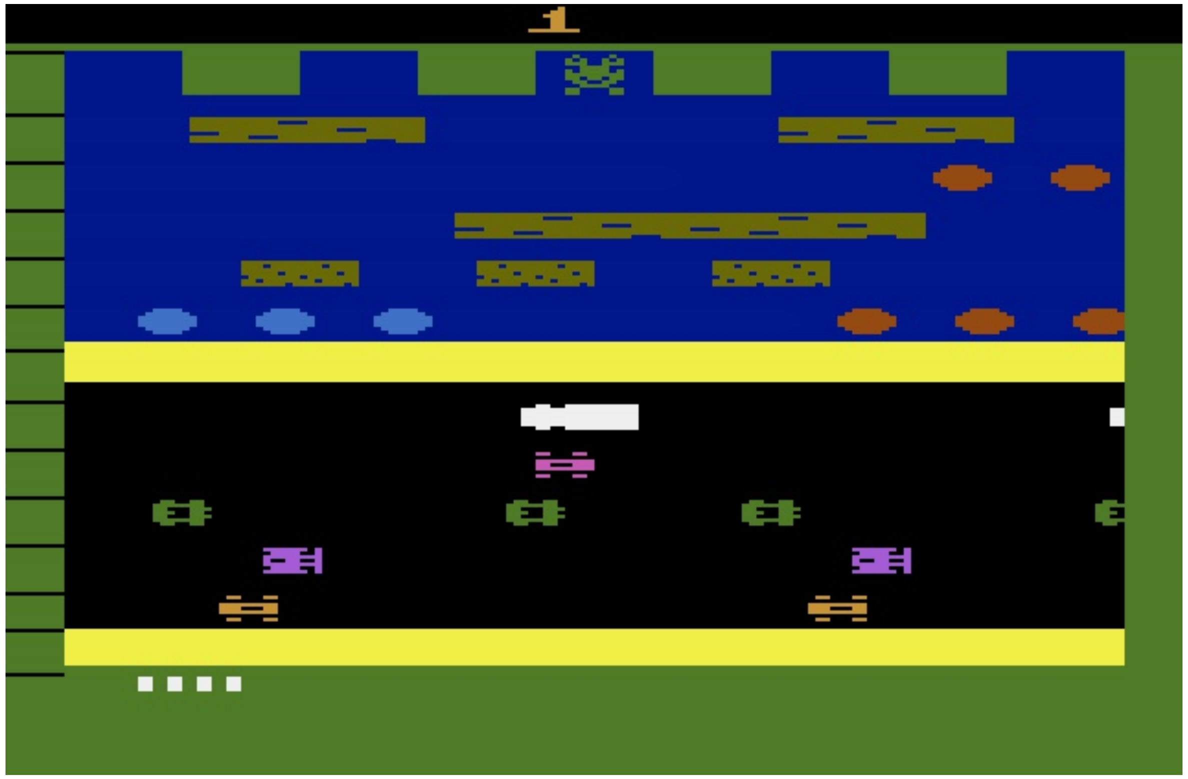
Starpath Supercharger


Supercharger™
BY
STARPATH™
CORPORATION
(formerly ARCADIA)
PATENT PENDING
Atari® and VCS™ are trademarks of ATARI, INC. STARPATH CORP. is not related to ATARI, INC.
©1982 Starpath™ (ROM Code & Custom Circuit)
• Plug the Supercharger into Atari® VCS™ (with power OFF).
• Plug Supercharger cable into earphone jack of cassette player.
• Switch power ON. Follow instructions on TV screen.

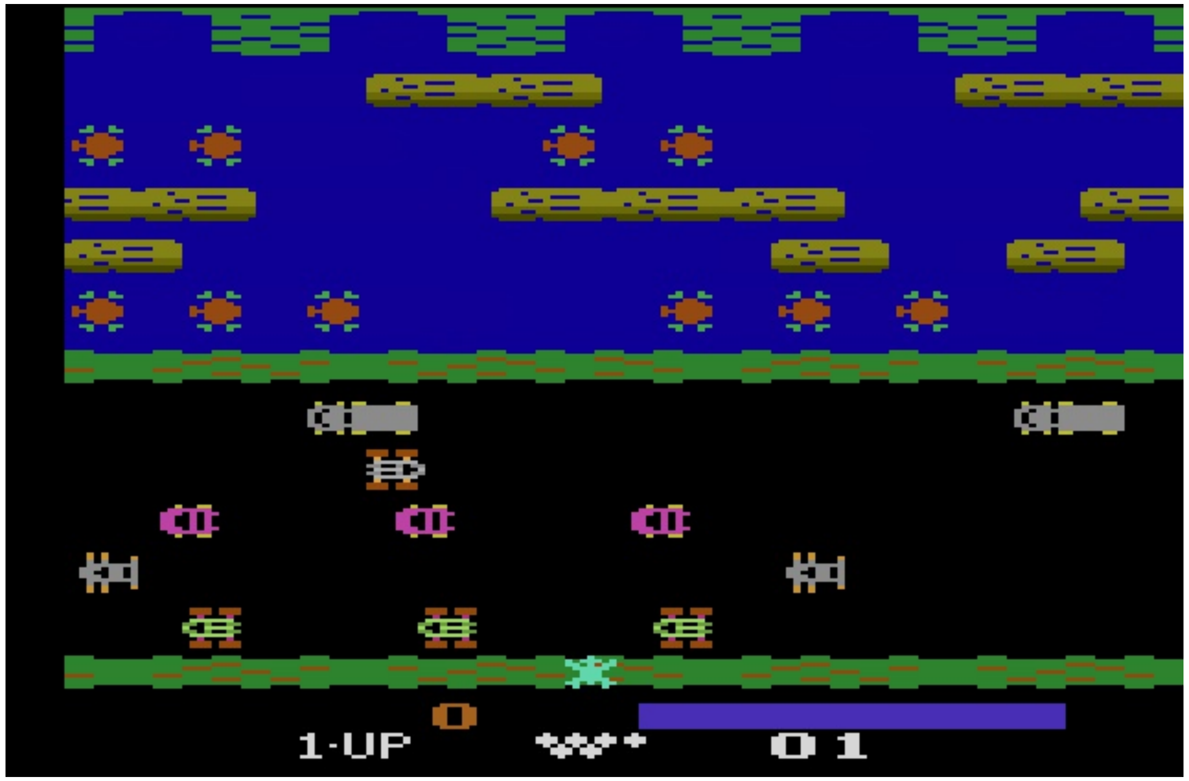
Starpath Supercharger

- Released in 1982 for $45
- Uses audio to load programs (games were sold on cassettes)
- 6 K RAM, 2 K BIOS
- Program MAKEWAV converts ROM image to sound file

Starpath Supercharger



Supercharger
BY
STARPATH
(formerly ARCADIA)
PATENT PENDING
Atari® and VCS™ are trademarks of ATARI, INC. STARPATH CORP. is not related to ATARI, INC.
©1982 Starpath™ (ROM Code & Custom Circuit)
• Plug the Supercharger into Atari® VCS™ (with power OFF).
• Plug Supercharger cable into earphone jack of cassette player.
• Switch power ON. Follow instructions on TV screen.



Krokodile Cartridge

Krokodile Cartridge

- Released in 2005 for $99
- Uses serial port to load programs
- 512 K Flash ROM, 32 K RAM

Harmony Cartridge

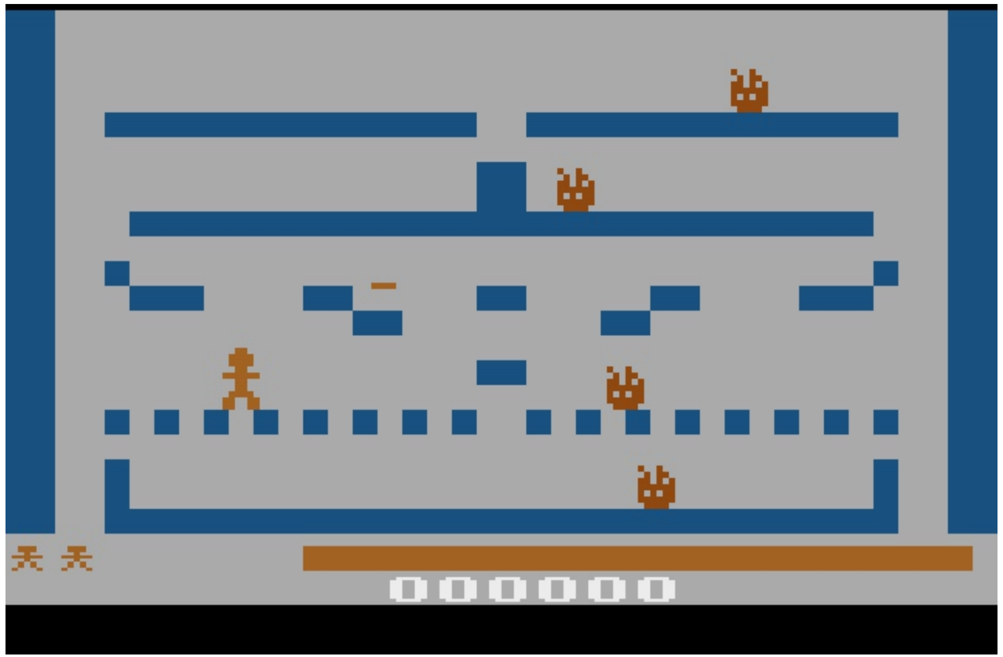
Harmony Cartridge

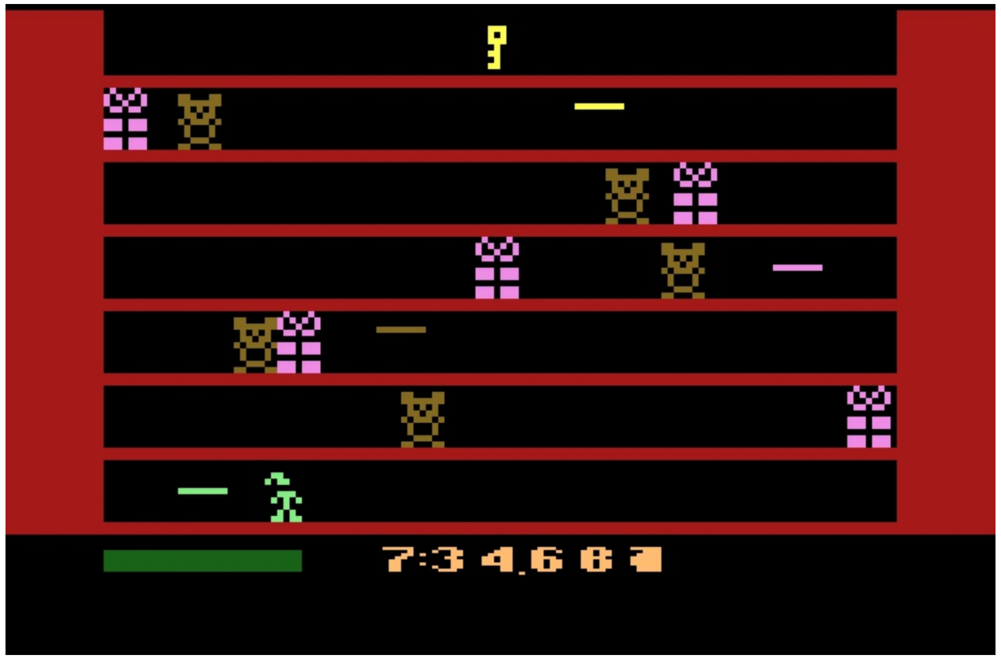
- Released in 2009
- Still produced, sells for $59.99 and $79.99 http://harmony.atariage.com
- Uses SD card or USB to load programs
- 32 K Flash ROM, 8 K RAM
- 70 MHz ARM processor
- Melody variation used by AtariAge to produce stand alone games

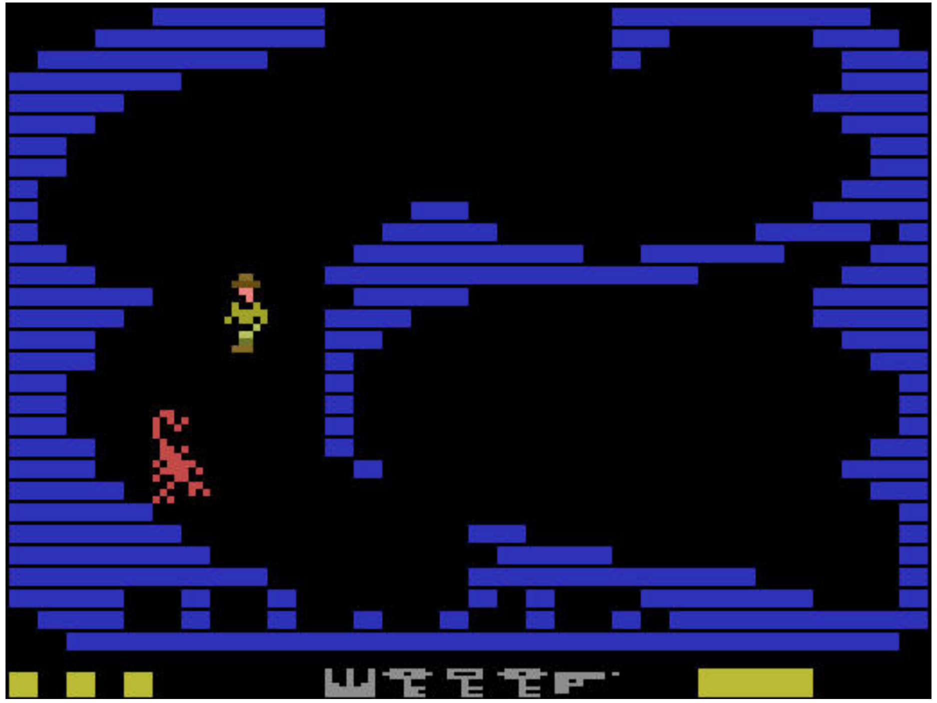
batari Basic

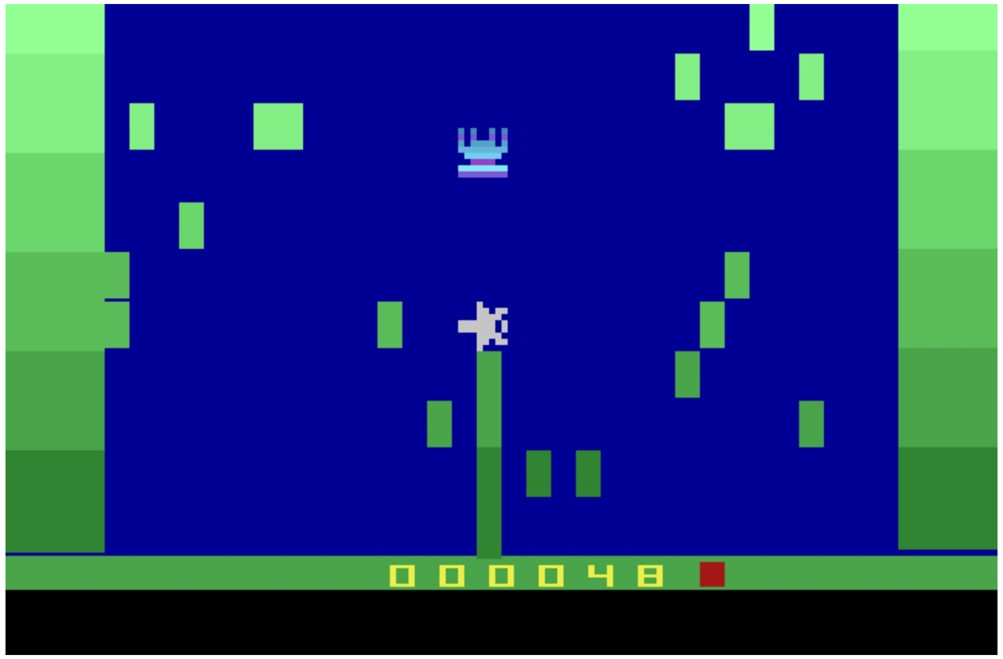
- Provides a simpler way to create Atari games
- Uses a BASIC like language for game logic
- Provides a number of prebuilt Kernels

batari Basic games



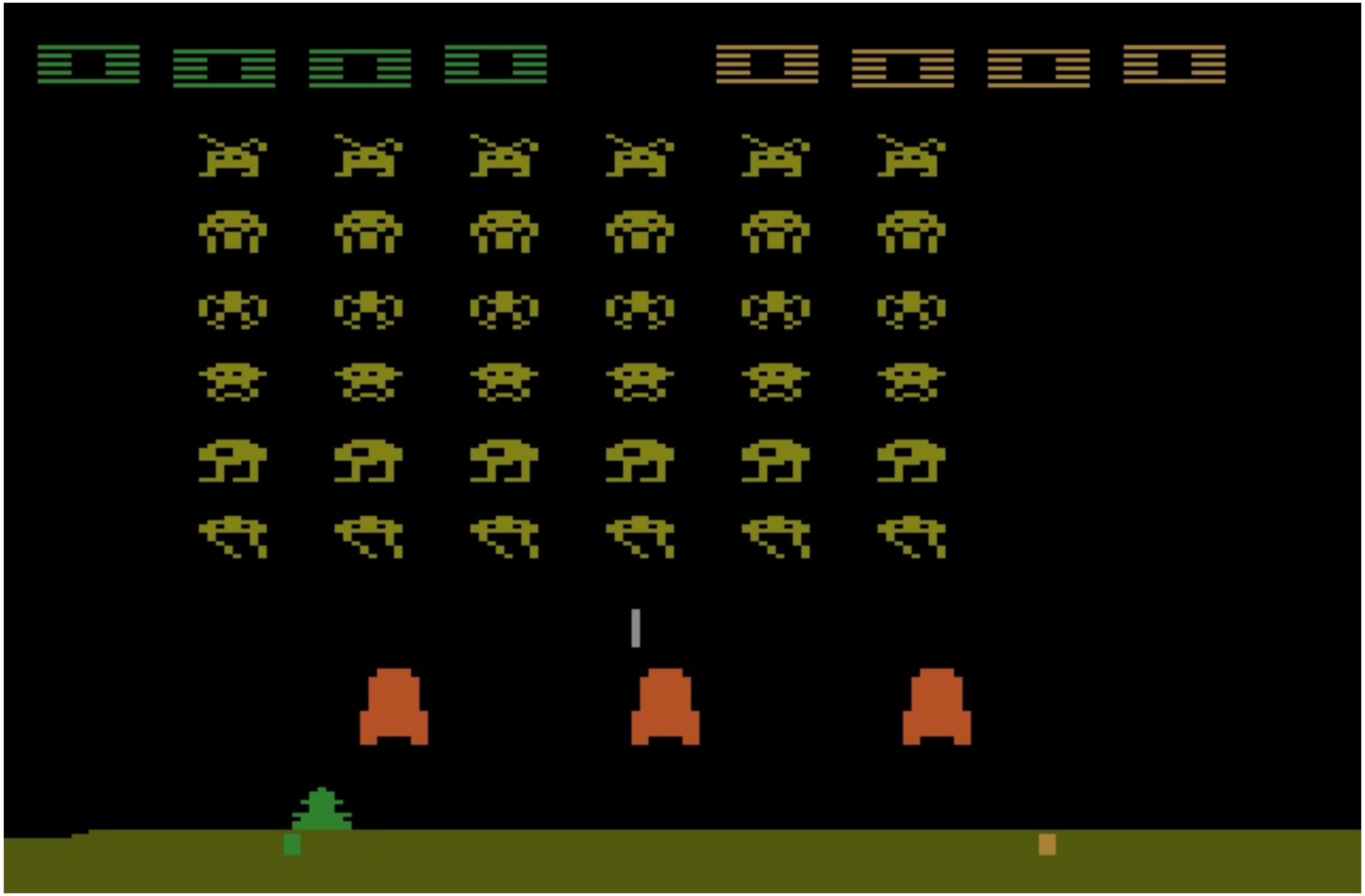






How are the limited objects used to create complex games?

Space Invaders



⬜ Background

□ HMOVE

🟪 Playfield

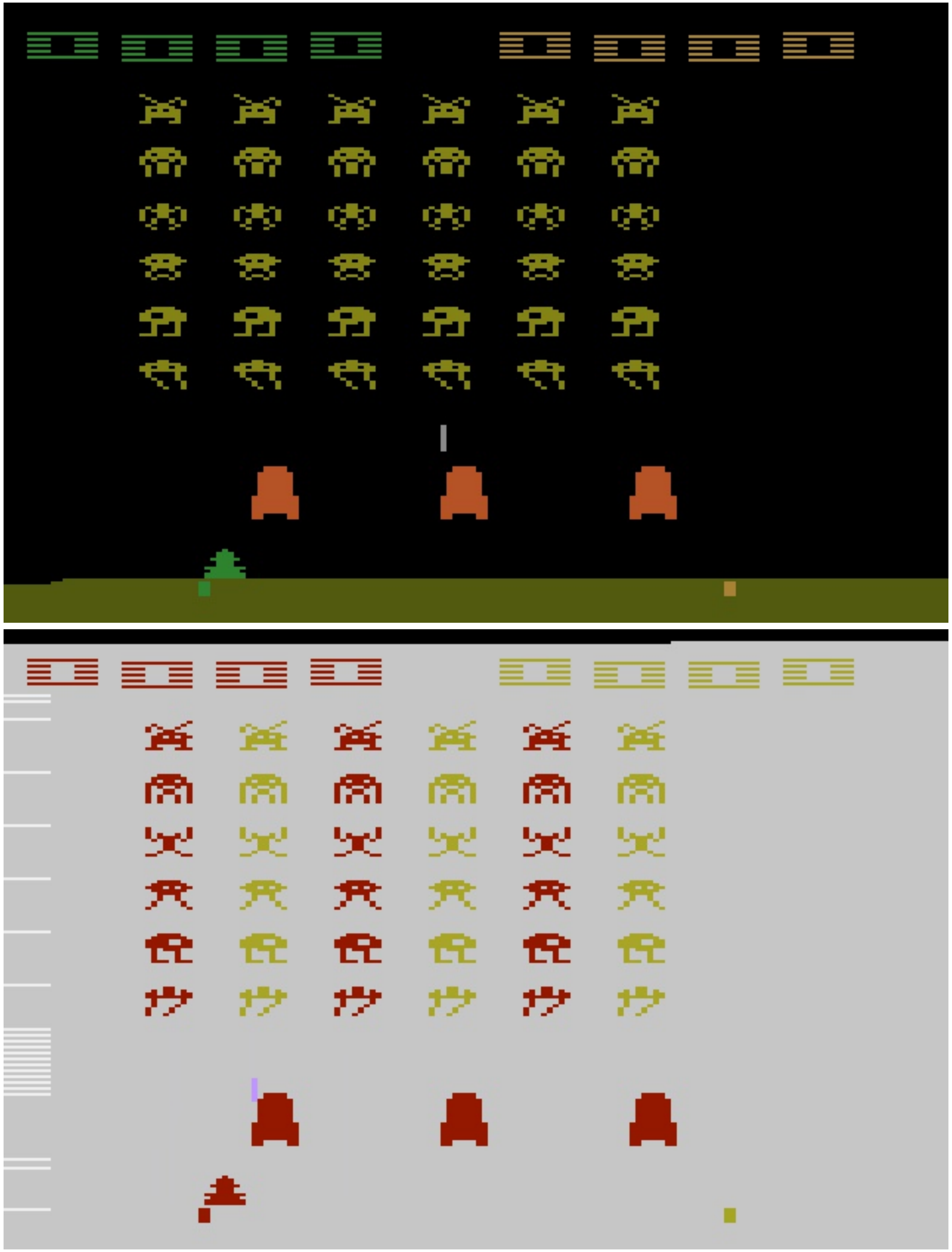
🟪 Ball

🟧 Player 0

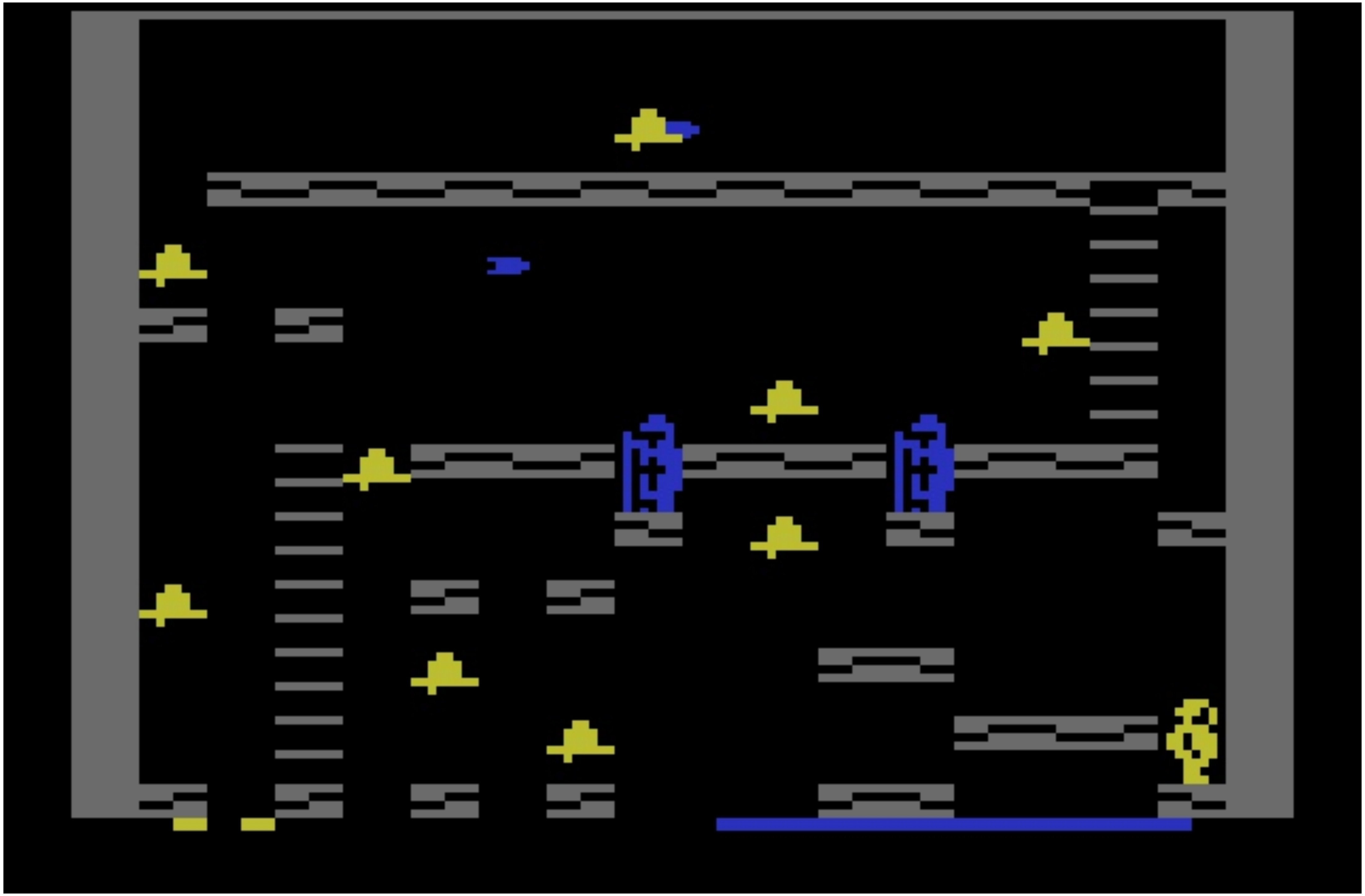
🟨 Player 1

🟧 Missile 0

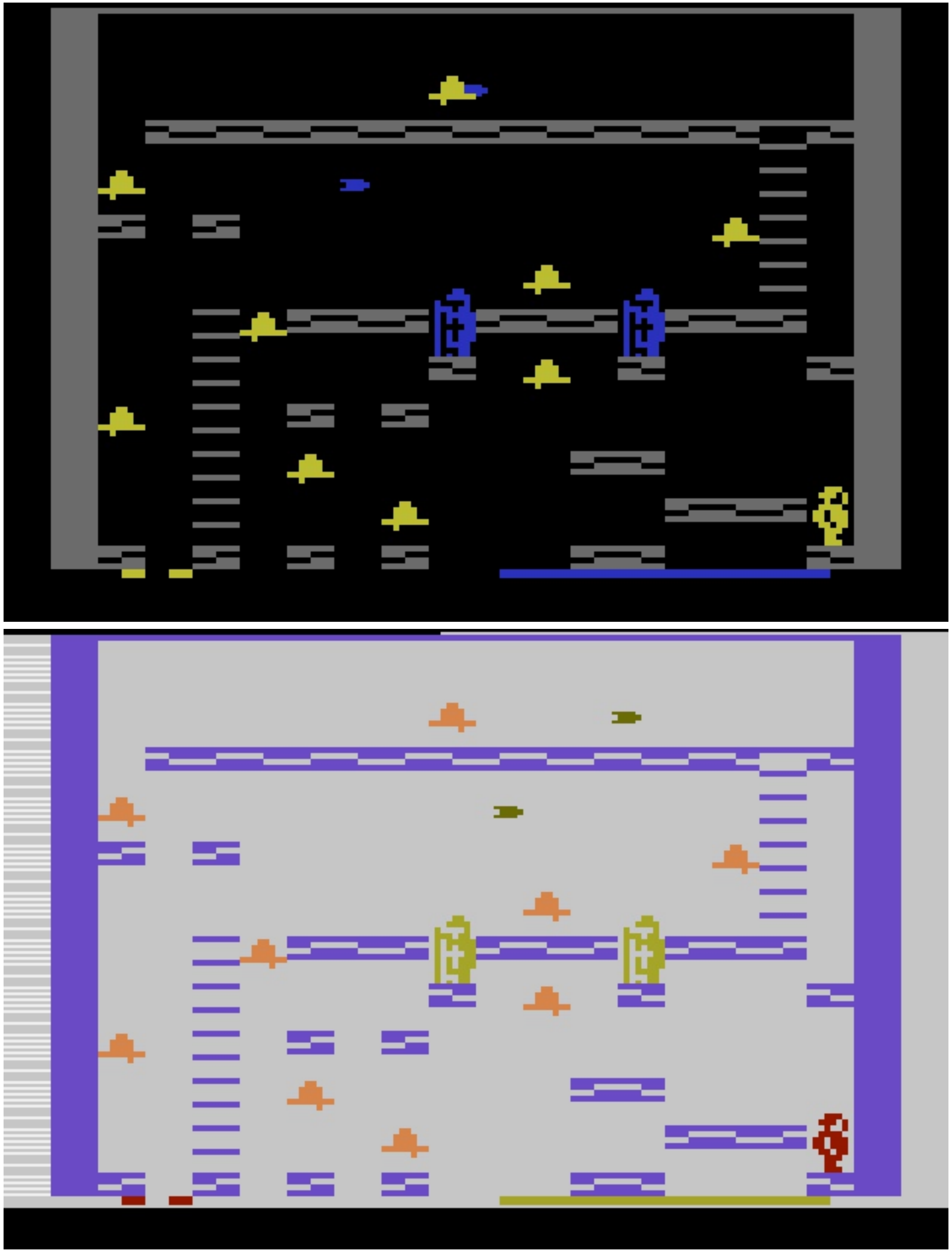
🟩 Missile 1



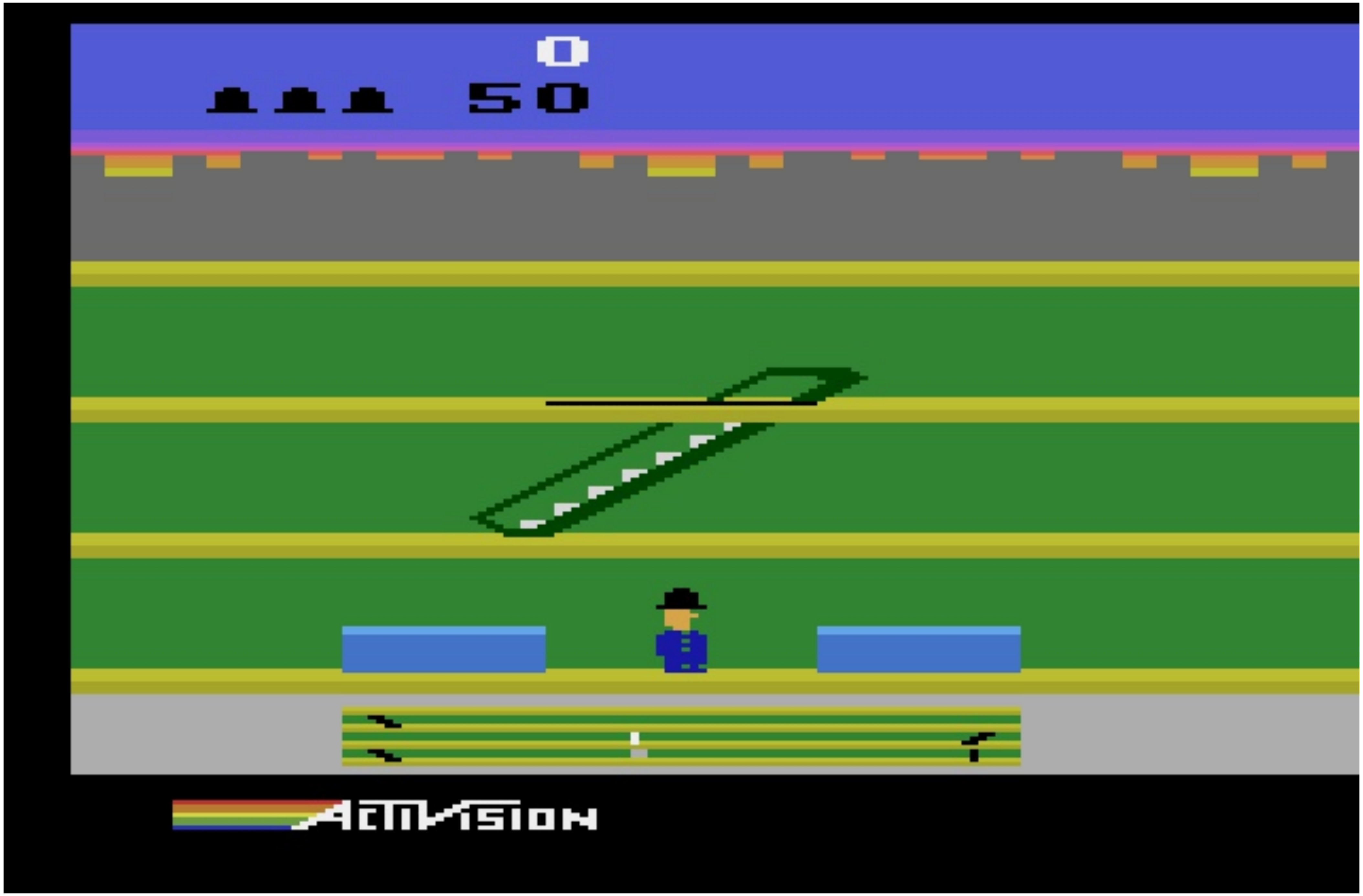
Hunchy II



- ⬜ Background
- □ HMOVE
- 🟪 Playfield
- 🟪 Ball
- 🟧 Player 0
- 🟨 Player 1
- 🟧 Missile 0
- 🟩 Missile 1



Keystone Kapers



⬜ Background

□ HMOVE

🟪 Playfield

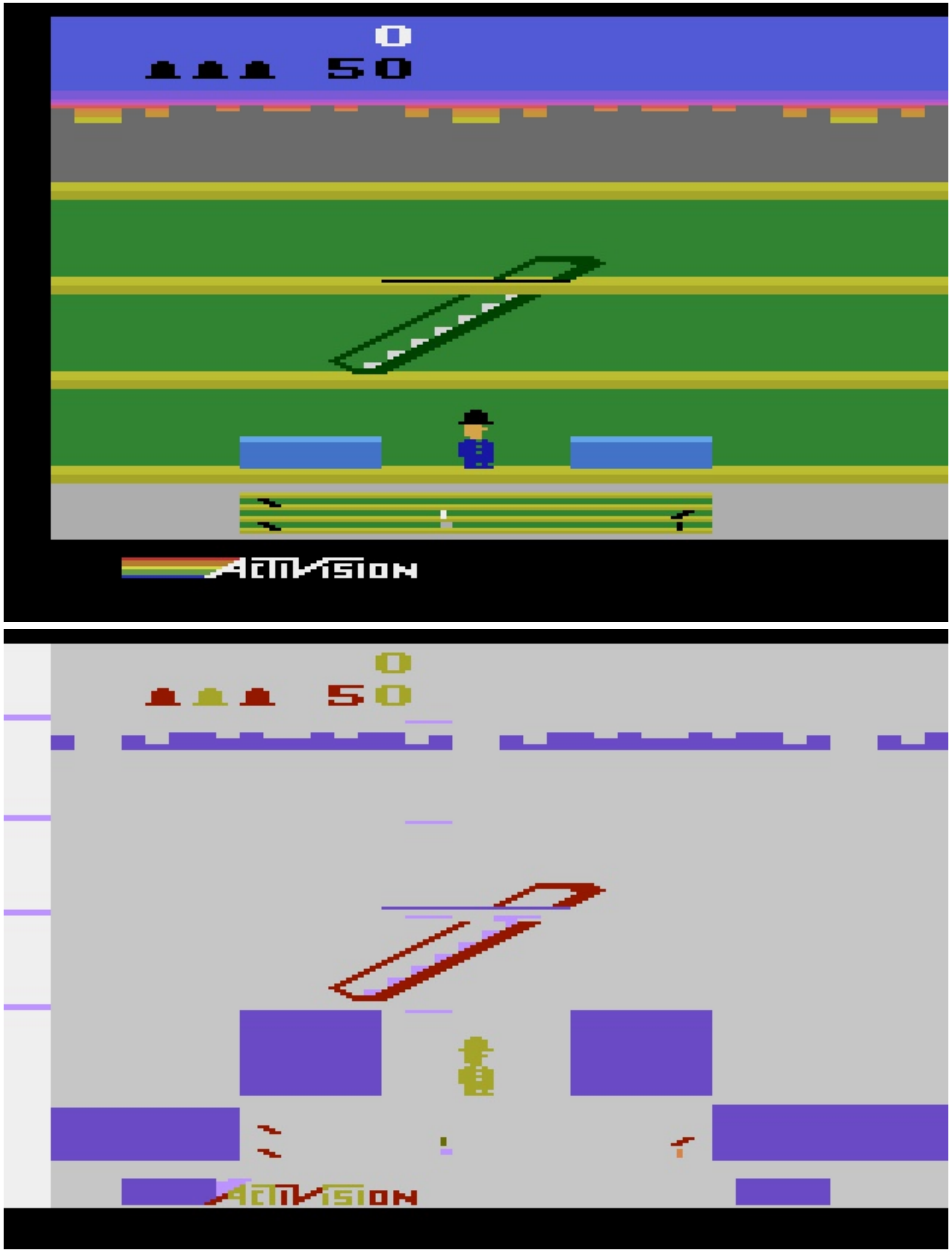
🟪 Ball

🟧 Player 0

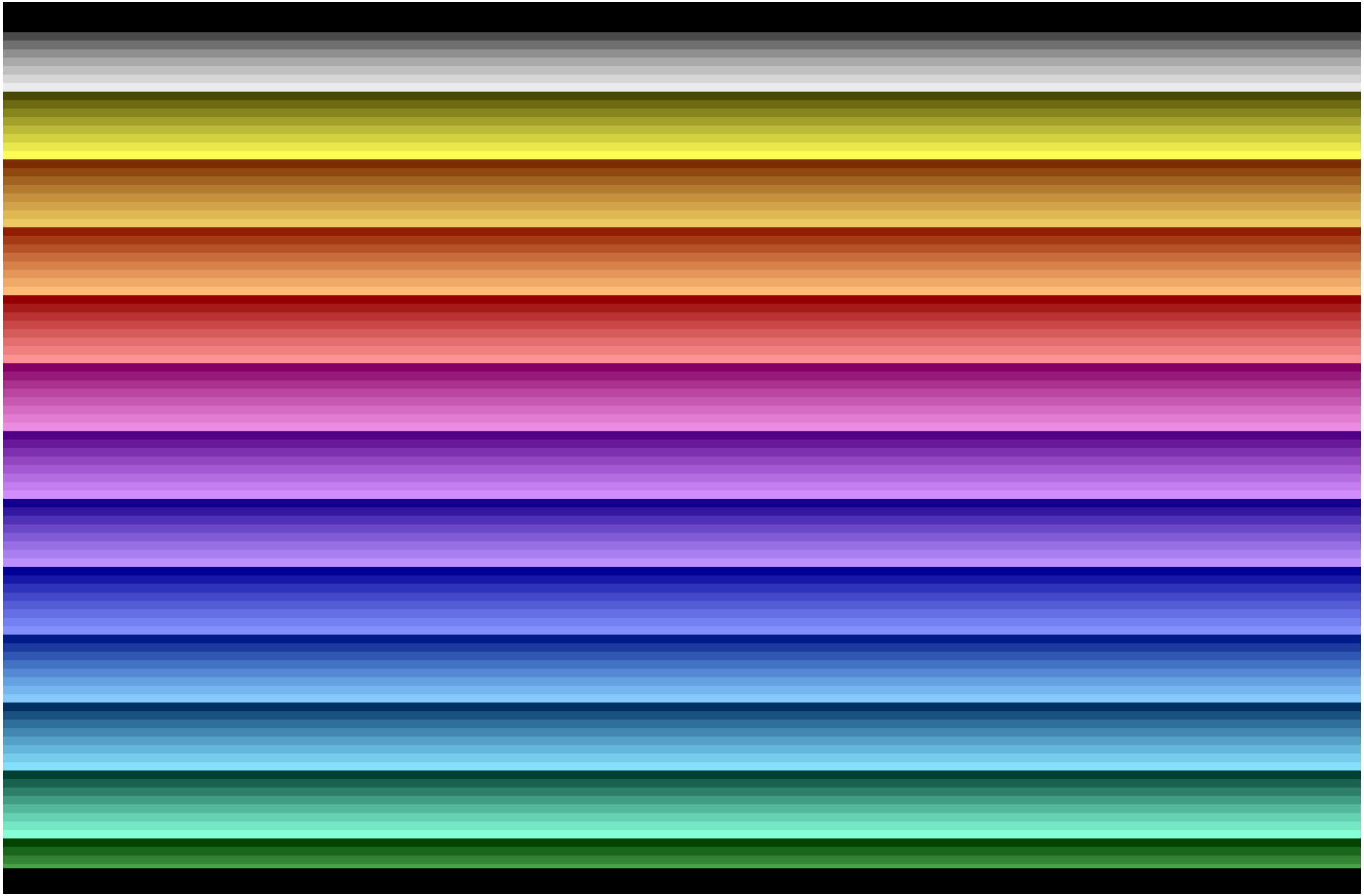
🟨 Player 1

🟧 Missile 0

🟨 Missile 1



Sample Program



Program Layout

- Initialize DASM
- Define RAM usage
- Define Start of Cartridge
- Initialize Atari
- Main Loop
- Define End of Cartridge

Initialize DASM

```
; tell DASM type of CPU
    PROCESSOR 6502

; vcs.h contains the standard definitions
; for TIA and RIOT registers
    include vcs.h

; macro.h contains commonly used routines
    include macro.h
```

Define RAM usage

```
; define a segment for variables
; .U means uninitialized, does not end up in ROM
    SEG.U VARS

; RAM starts at $80
    ORG $80

; holds background color for first scanline of frame
BackgroundColor: ds 1     ; stored in $80

; holds playfield color for first scanline of frame
PlayfieldColor:  ds 1     ; stored in $81

; holds # of scanlines left for the kernel to draw
LineCount:       ds 1     ; stored in $82
```

Define Start of Cartridge

```
; define a segment for code
    SEG CODE

; ROM starts at $F000
    ORG $F000
```

Initialize Atari

```
InitSystem:

; CLEAN_START is a macro found in macro.h
; it sets all RAM, TIA registers
; and CPU registers to 0
    CLEAN_START

; for sample program, this sets playfield
; to output as vertical stripes
    lda #$AA
    sta PF0
    sta PF2
    lda #$55
    sta PF1
```

Main Loop

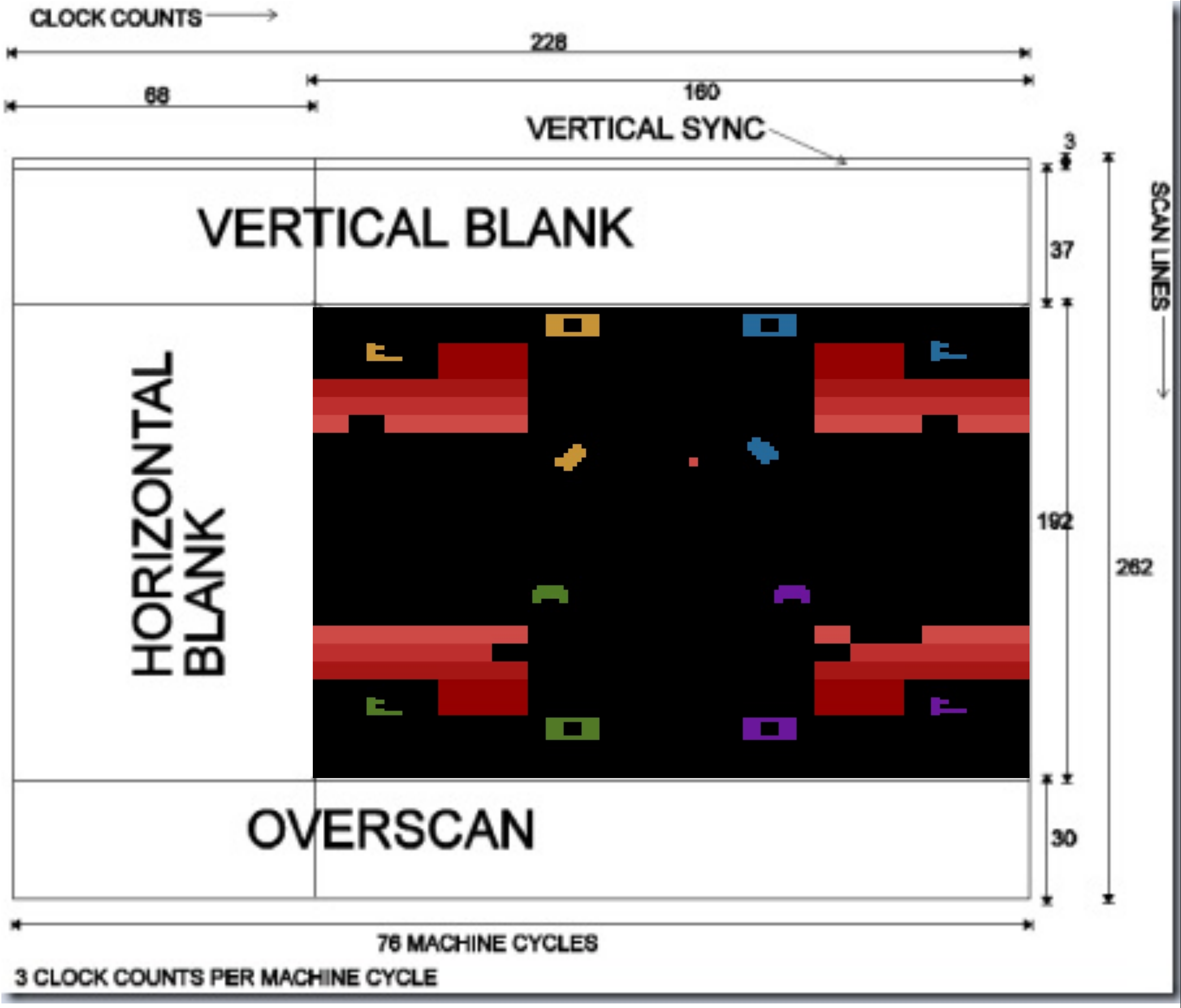
Sync signal

Game logic

Kernel
(draw display)

Game logic

Repeat Loop



Sync Signal

```
VerticalSync:
    lda #2
    sta WSYNC
    sta VSYNC   ; turn on Vertical Sync signal
    sta VBLANK  ; turn on Vertical Blank signal
    lda #47
    sta TIM64T  ; set timer for end of Vertical Blank
    sta WSYNC   ; 1st scanline of Sync Signal
    sta WSYNC   ; 2nd scanline of Sync Signal
    lda #0
    sta WSYNC   ; 3rd scanline of Sync Signal
    sta VSYNC   ; turn off Vertical Sync signal
```

Vertical Blank

```
VerticalBlank:
;--------------------------
;  game logic starts here
;--------------------------
    inc BackgroundColor
    dec PlayfieldColor
    lda #199
    sta LineCount
;--------------------------
;   game logic ends here
;--------------------------

VBwait:
    sta WSYNC
    bit TIMINT
    bpl VBwait   ; loop until the timer ends
```

Kernel

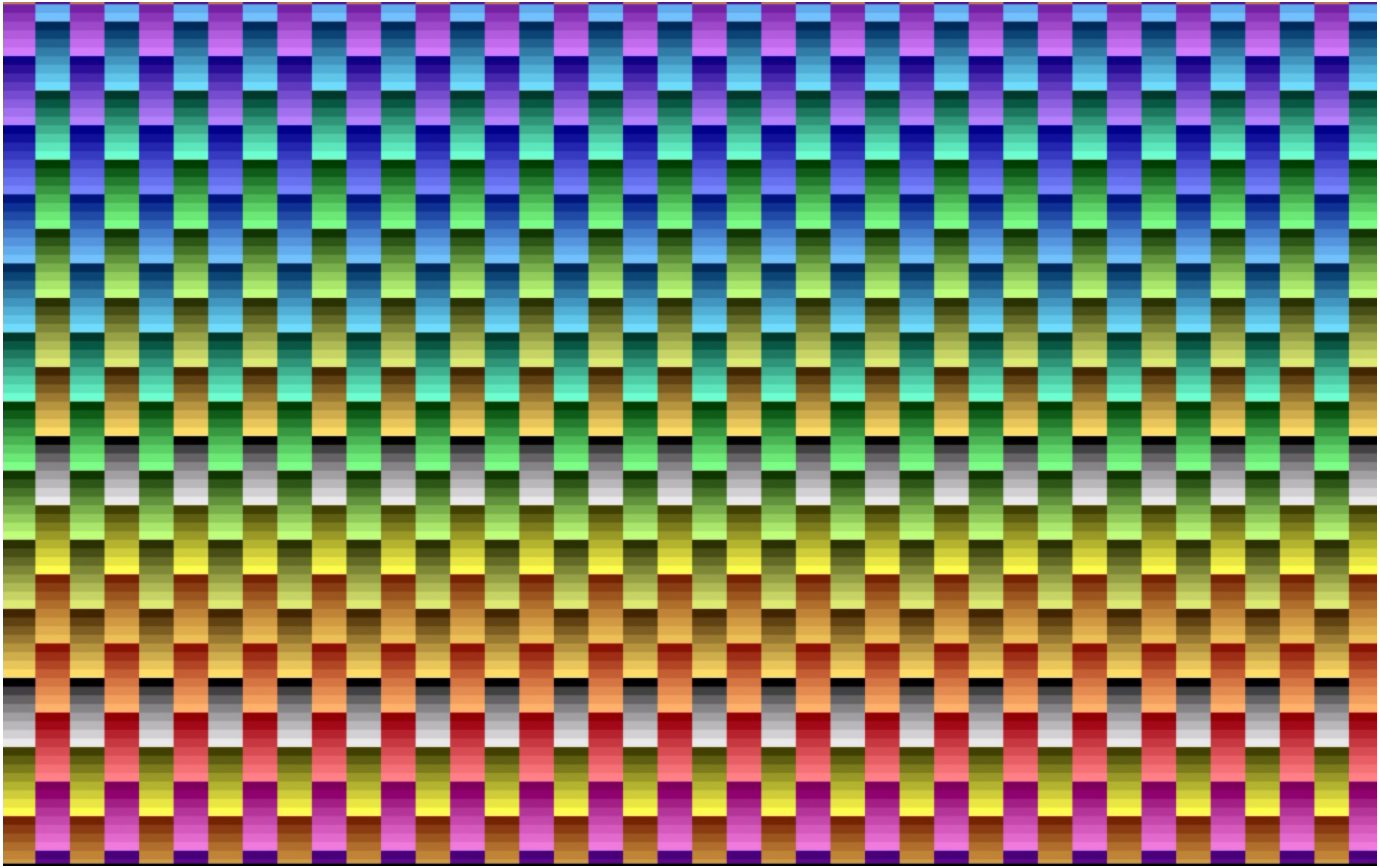
```
    sta WSYNC
    lda #0
    sta VBLANK
    sta COLUBK         ; color first scanline black
    sta COLUPF         ; color first scanline black
    ldx BackgroundColor
    ldy PlayfieldColor

KernelLoop:
    sta WSYNC
    stx COLUBK         ; update background color
    sty COLUPF         ; update playfield color
    inx                ; change X for next scanline
    iny                ; change Y for next scanline
    dec LineCount
    bne KernelLoop
```

Overscan

```
OverScan:
    sta WSYNC
    lda #2
    sta VBLANK       ; turns video output off
    lda #23
    sta TIM64T       ; set timer for end of Overscan

;-----------------------------------
;  additional game logic goes here
;-----------------------------------

OSwait:
    sta WSYNC
    bit TIMINT
    bpl OSwait       ; loop until the timer ends

    jmp VerticalSync ; start the next frame
```

Define End of Cartridge

```
; set destination of 6507 Interrupt Vectors
    ORG $FFFA
    .WORD InitSystem ; NMI
    .WORD InitSystem ; RESET
    .WORD InitSystem ; IRQ and BRK
```



Resources

Atari Age	http://www.atariage.com/
Mini dig	http://www.qotile.net/minidig/
Stella	http://stella.sourceforge.net/
Harmony	http://harmony.atariage.com/
Dasm	http://dasm-dillon.sourceforge.net/
Atari 2600 Programming	http://www.atariage.com/forums/forum/50-atari-2600-programming/
2600 Programming for Newbies	http://www.atariage.com/forums/forum/31-2600-programming-for-

batari Basic

batari Basic	http://bataribasic.com/
Atari Age forum	http://www.atariage.com/forums/forum/65-batari-basic/
Random Terrain	http://www.randomterrain.com/atari-2600-memories-batari-basic-commands.html
Visual bB	http://www.atariage.com/forums/topic/123849-visual-bb-1-0-a-new-ide-for-batari-basic/

Questions?

Presentation will be made available at

http://www.spiceware.org