The Atari 2600
Video Computer System

The Ultimate Talk

The history, the hardware
and how to write programs

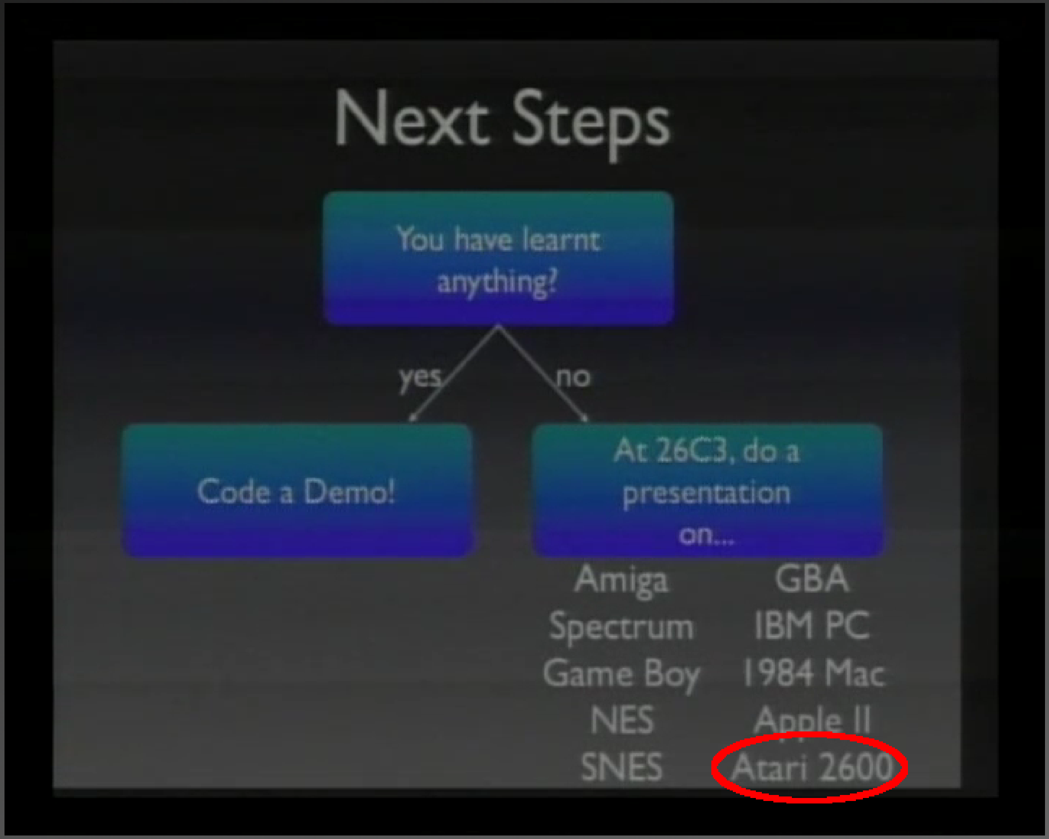
by Sven Oliver ('SvOlli') Moll

28c3 - Behind enemy lines - 2011-12-27 - 12:45 - Saal 3

Motivation for this talk (1)

The motivation came from two different aspects

Michael Steil's talks about the C=64 and 6502 inspired me to start a talk about retro computing



Next Steps

You have learnt anything?

yes

no

Code a Demo!

At 26C3, do a presentation on...

Amiga

Spectrum

Game Boy

NES

SNES

GBA

IBM PC

1984 Mac

Apple II

Atari 2600

Motivation for this talk (2)

Why the Atari 2600 Video Computer System?

Or better: why start coding on the 2600 today?

The CPU is well known and very well documented

The video chip is too, and it differs from all others

I read the programmer's manual and thought: "Wow, the 2600 is the most f***ed up 6502 compatible system I've ever seen, I've got to give this one try!"

Acknowledgements

The Atari 2600 has a huge homebrew scene running since the 90's

I learned a lot from other people, who pioneered homebrew on the Atari 2600:

Fred Quimby, Thomas Jentzsch, Paul Slocum, Duane Alan Hahn, Manuel Polik, Eckhard Stolberg, Andrew Davie, Ed Federmeyer, Glenn Saunders, Nukey Shay, Chris Wilkson, Erik Mooney and many others.
Sorry I've forgotten to include your name!

Thanks

Thanks to the following sites for providing me with information, supporting me and / or letting me use their content for this talk

http://www.alienbill.com/2600/
http://www.atariage.com/
http://www.atarimania.com/
http://www.biglist.com/lists/stella/
http://www.ccmuseum.de/
http://www.qotile.net/minidig/
http://www.randomterrain.com/
http://en.wikipedia.org/wiki/Atari_2600

Part 1:

The history

Atari history

Founded 1972 by Nolan Bushnell and Ted Dabney

Best known for the arcade hit "Pong" (1972)

Recognized as the first worldwide popular video game, though it was not the first overall (http://en.wikipedia.org/wiki/First_video_game)

Same for the Atari 2600 VCS (1977)

Between 1972 and 2001 Atari released several well-known arcade games, several of them were re-implemented for the Atari 2600

Design history (1)

Atari's first home release was "Home Pong"

In 1975, Atari decided to produce a home game console based on a programmable design

Code named "Stella" after the bicycle of an engineer

3 processor designs were considered:

– Intel 8080

– Motorola 6800

– MOS 6502 (bought by Commodore before release)

Price was one of the key issues, should be cheap

Design history (2)

The basic design was set up in two days by a core engineering team together with Chuck Peddle of MOS

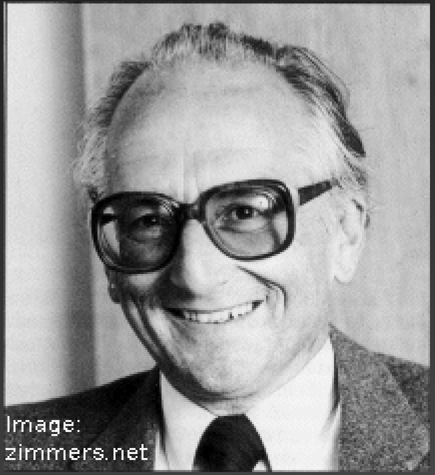


Image: zimmers.net

CPU and chipset were off-the-shelf components

Price for CPU + chipset was $12
(Intel and Motorola: $150 - $200)

A week after Motorola learned that they didn't get the deal, they sued MOS for patent infringements

Design history (3)

Chip for video and audio was still needed

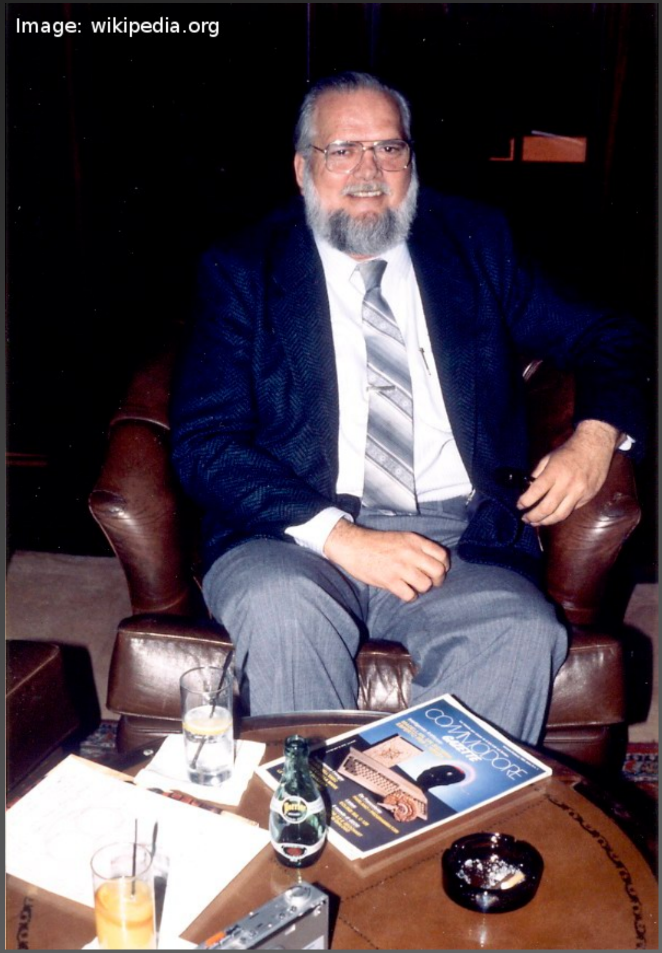


Image: wikipedia.org

Nothing suitable was available

Designed by Jay Miner

Using breadboard technology

Expensive design phase

Finished design was transferred into a chip, that chip was cheap to produce

Named "Television Interface Adapter (TIA)"

Breadboard example

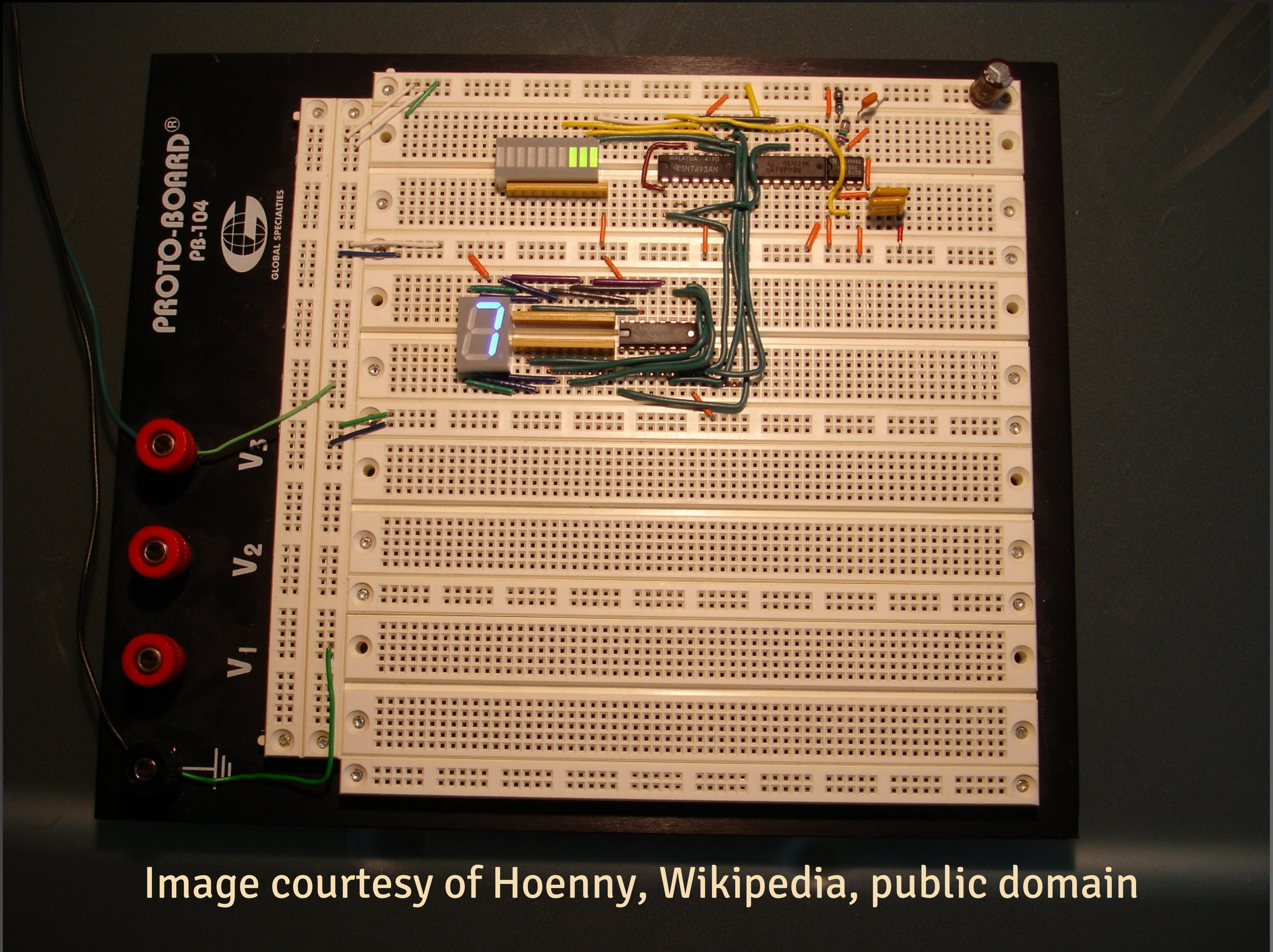


Image courtesy of Hoenny, Wikipedia, public domain

TIA die shot

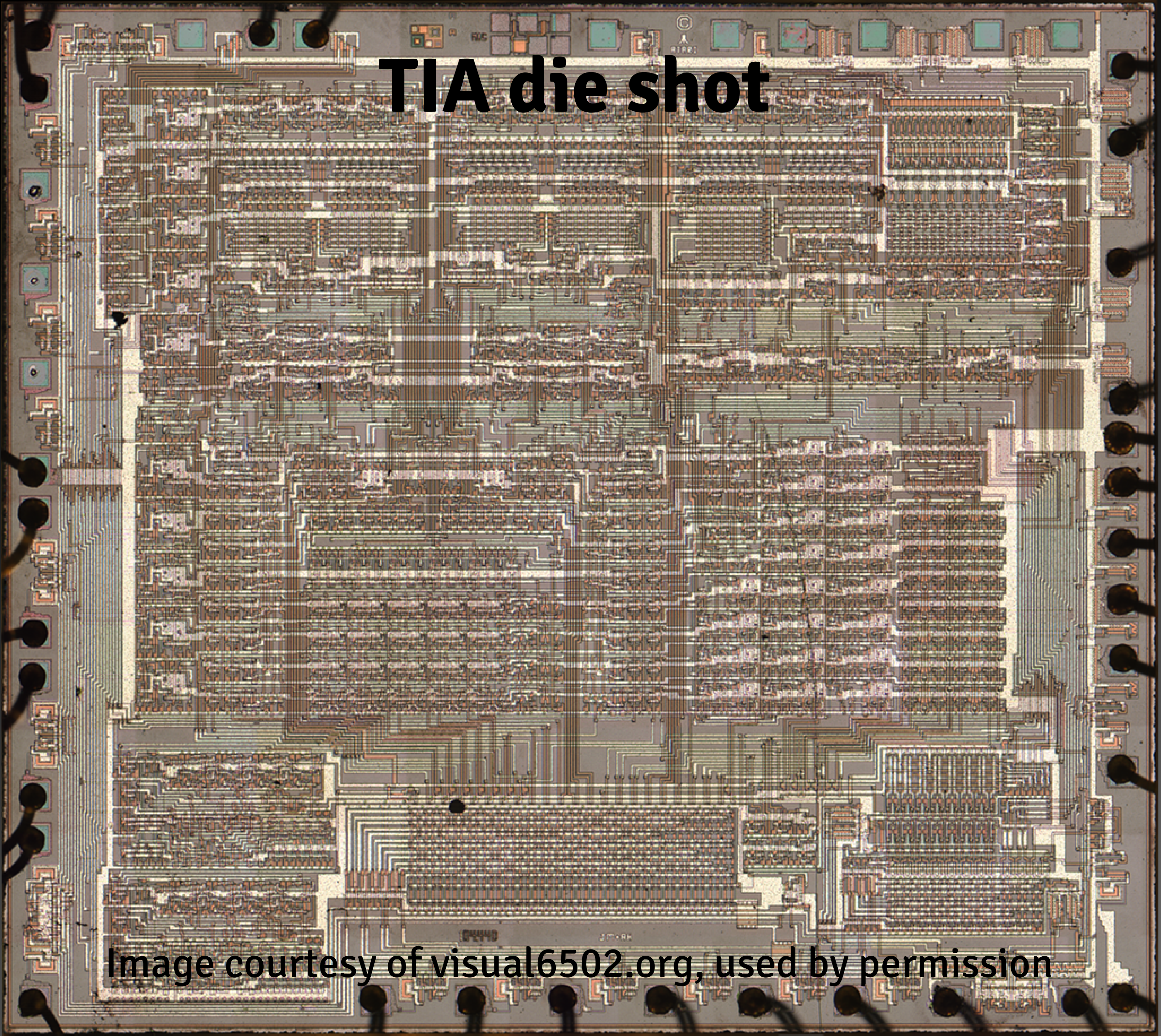


Image courtesy of visual6502.org, used by permission

Part of pop culture

Though not first on market, first home video game system that achieved broad distribution

Released in 1977, constantly revised, both internally and in appearance, still being 100% compatible

At the beginning of the 80's "Atari" was a synonym for home video gaming

Discontinued at the end of 1991

Still remains the game console that has been the longest in production, with a games catalog of 500+ different games in estimated 10000+ variations

Revision Overview (1)

6 switch model, wood design (1977) (PAL: 1978)

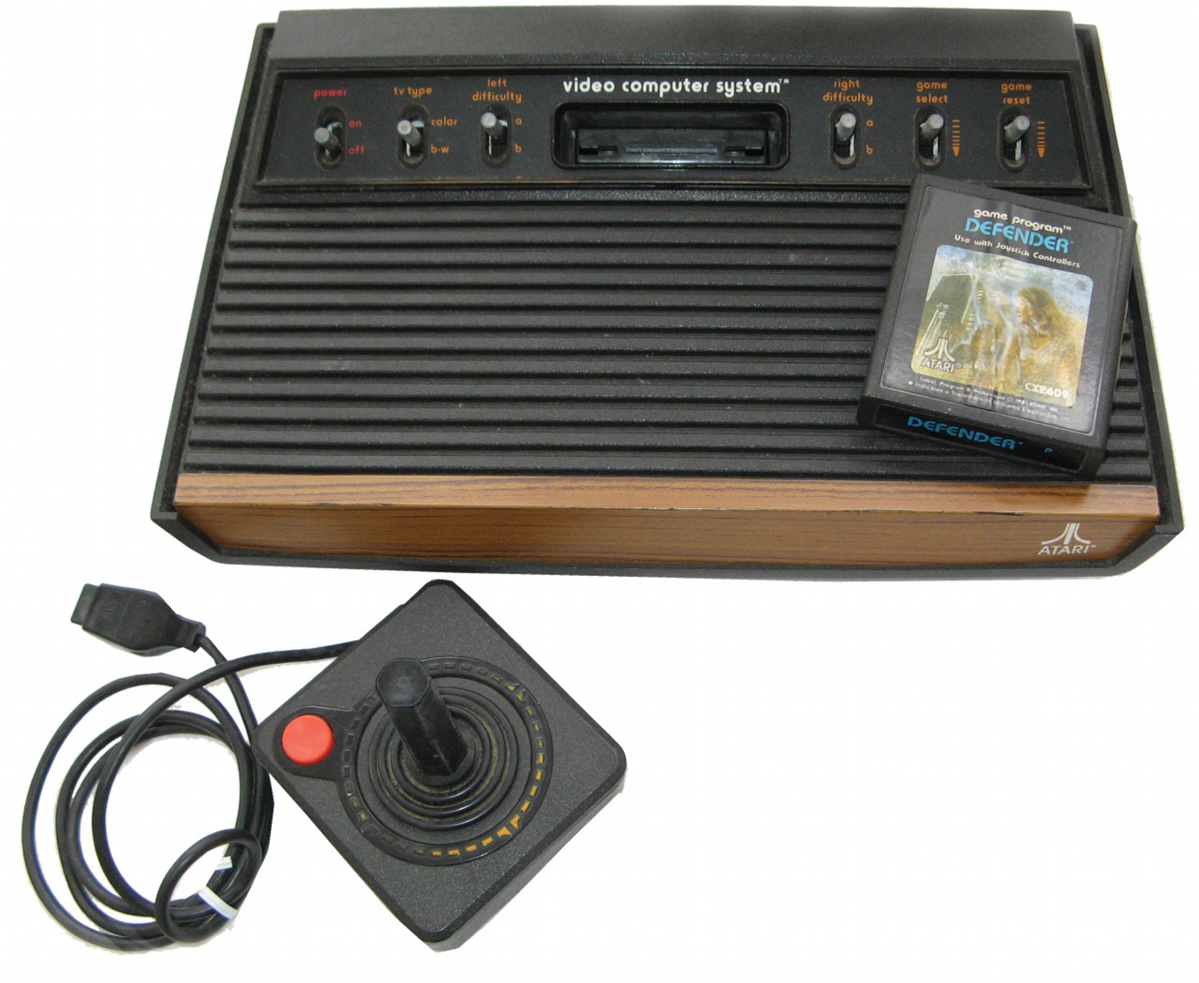


Image courtesy of www.ccmuseum.de, used by permission

Revision Overview (2)

4 switch model, wood design (1980)

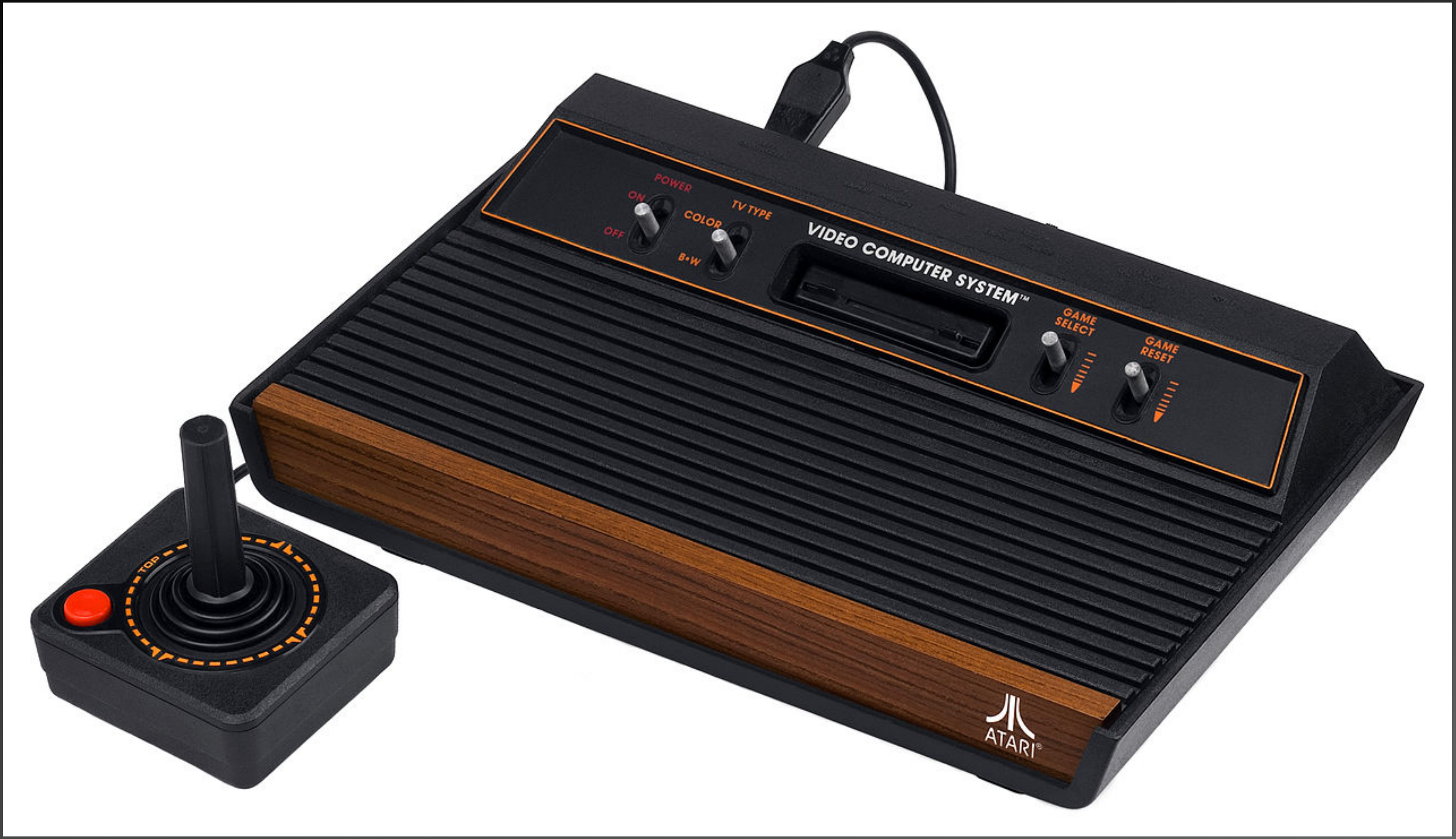
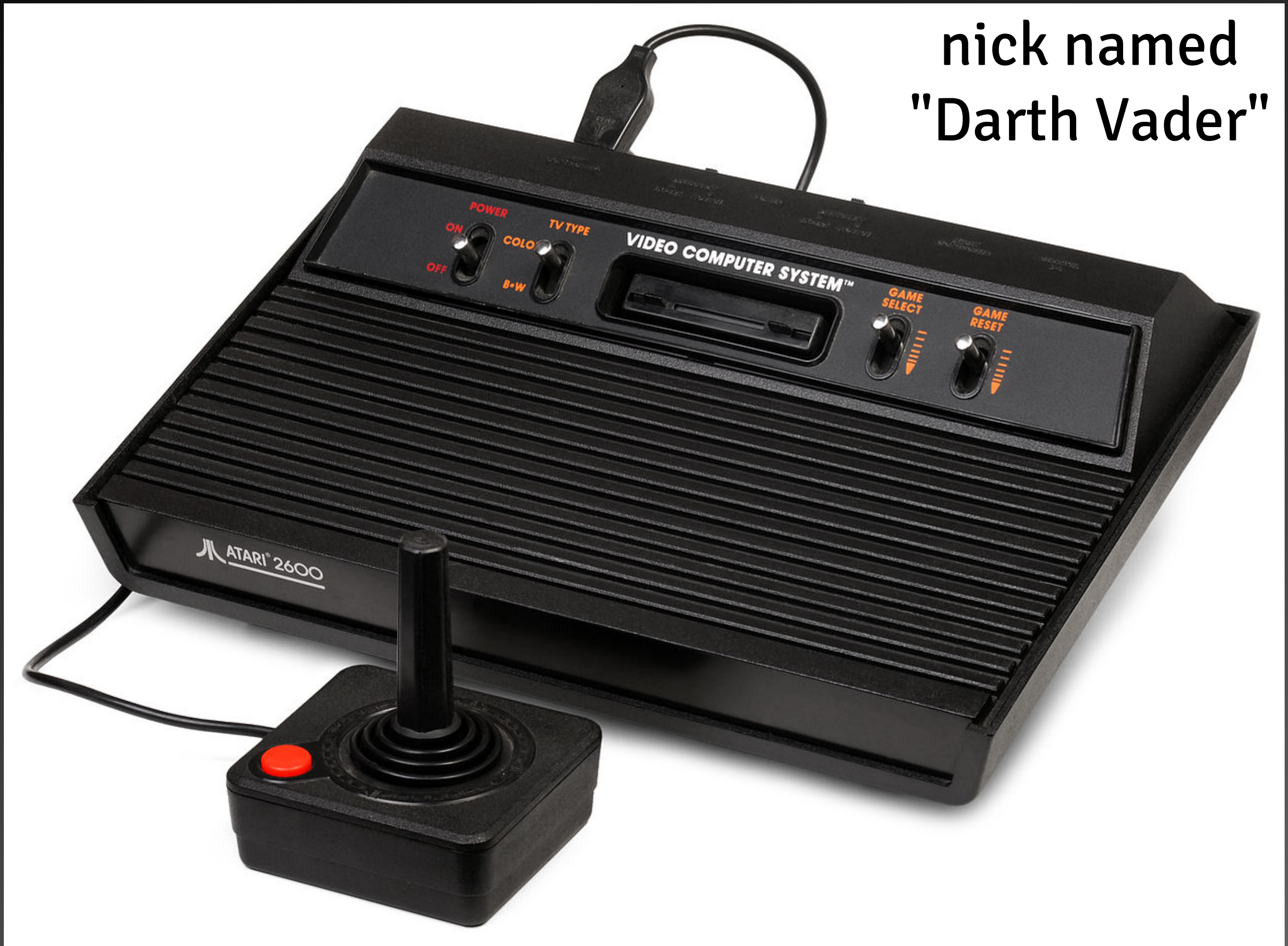


Image courtesy of Ewan-Alan, Wikipedia, public domain

Revision Overview (3)

4 switch model, black design (1982)



nick named "Darth Vader"

Image courtesy of Ewan-Alan, Wikipedia, public domain

Revision Overview (4)

Atari 2600 Jr (1984)

Image courtesy of Ewan-Alan, Wikipedia, public domain

Revision Overview (5)

Flashback 2+ (2: 2005, 2+: 2010)

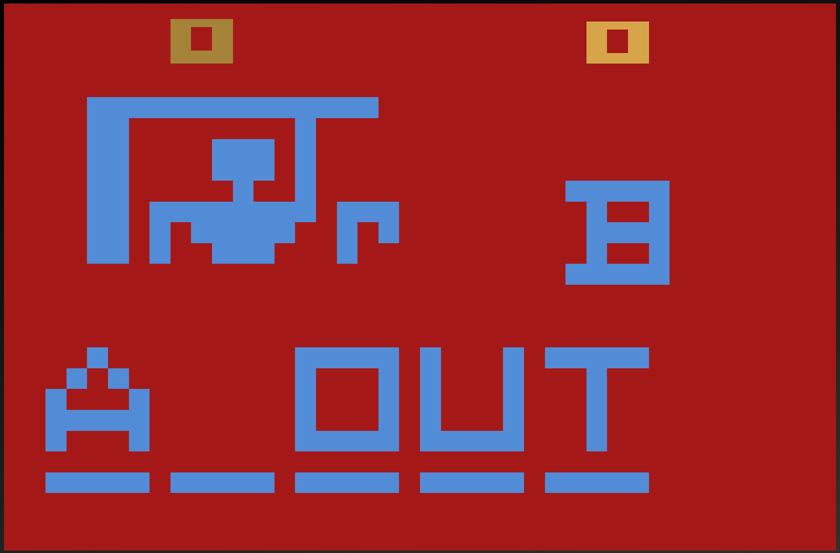
Flashback 2+ shares the same hardware,
but has a slightly different games collection

Inspirations for games (1)

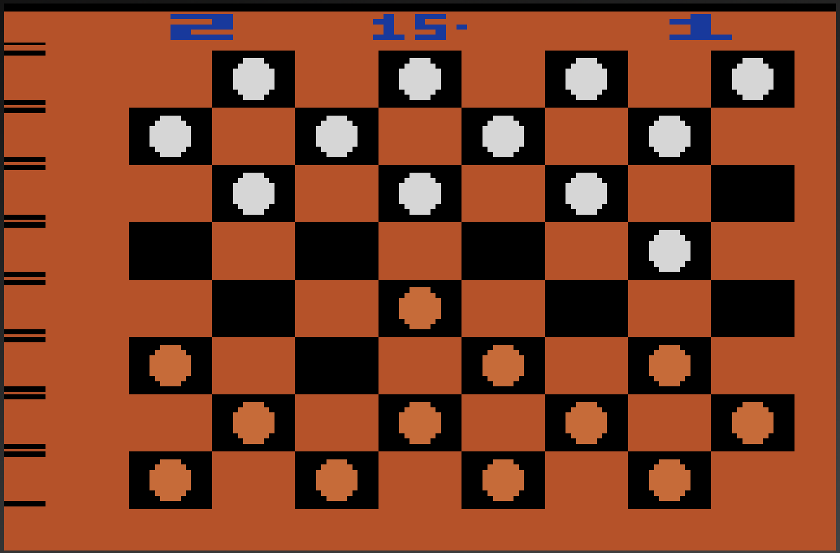
"Analog" games (board games, etc.)

- 3D Tic-Tac-Toe (Atari, 1978)
- Casino (Atari 1978)
- Hangman (Atari, 1978)
- Othello (Atari, 1978)
- Slot Machine (Atari, 1979)
- Video Checkers (Atari, 1978)
- Video Chess (Atari, 1978)
- Video Pinball (Atari, 1981)

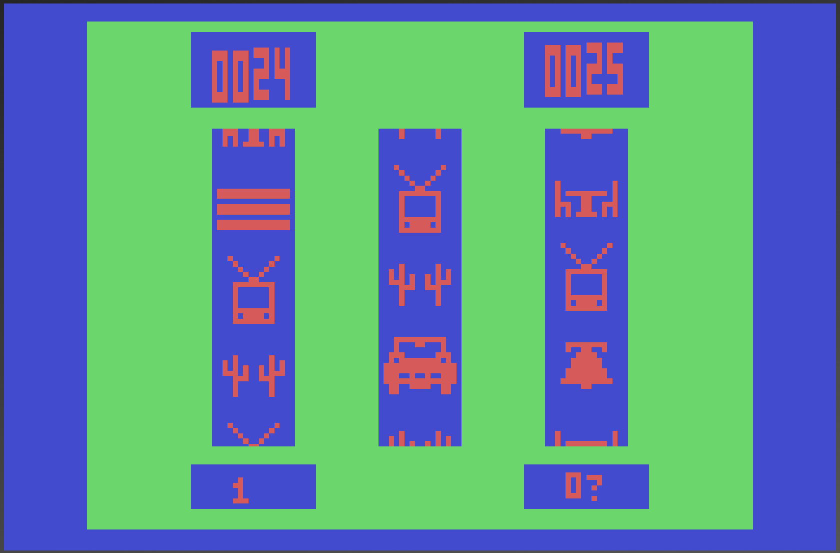
Game impressions (1)



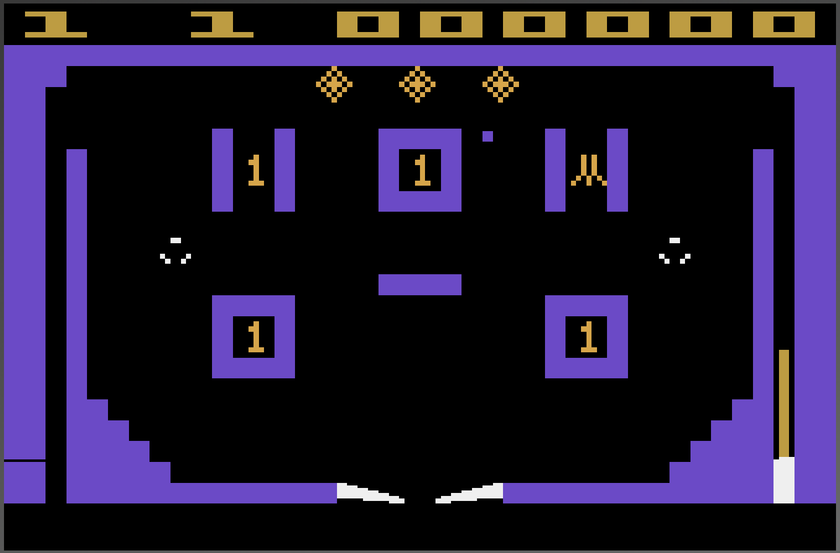
Hangman (Atari, 1978)



Video Checkers (Atari, 1978)



Slot Machine (Atari, 1979)



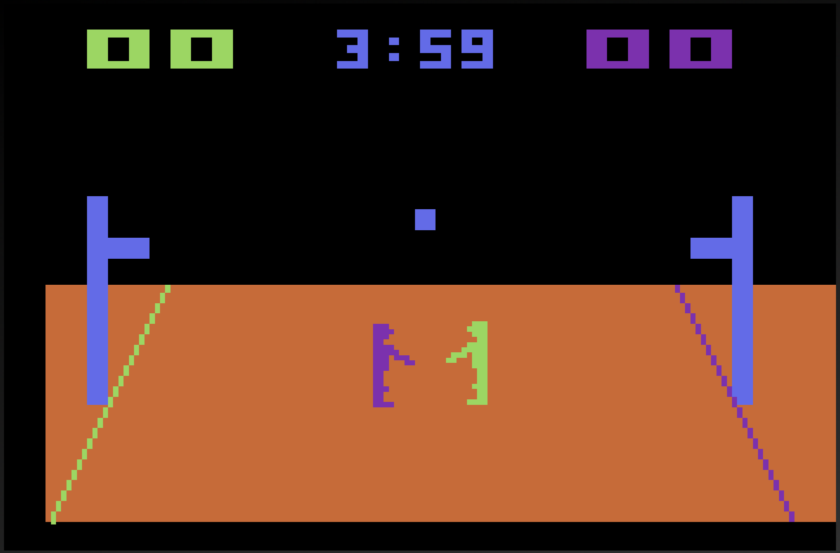
Video Pinball (Atari, 1981)

Inspirations for games (2)

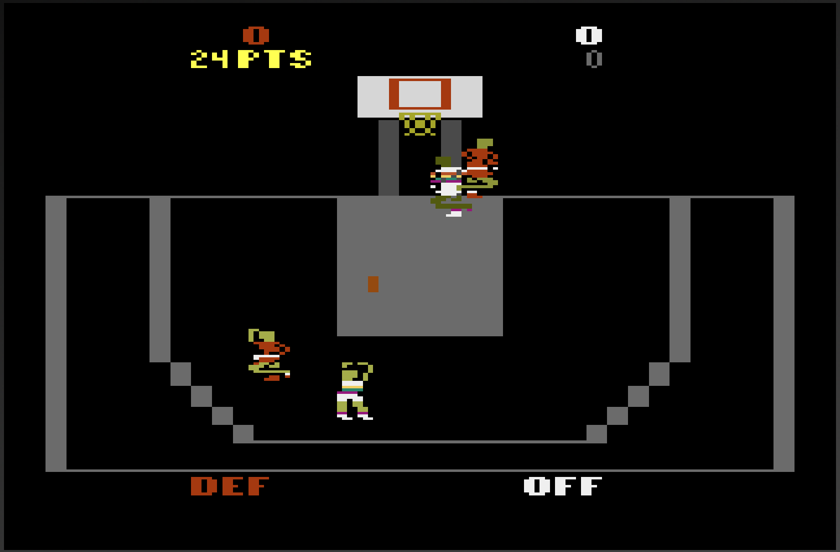
Sports games

- Basketball (Atari, 1978)
- Boxing (Activision, 1981)
- Bowling (Atari, 1978)
- Decathlon (Activision, 1983)
- Double Dunk (Atari, 1989)
- Polo (Atari 1978)
- Pelé's Soccer (Atari, 1981)
- Real Sports Soccer (Atari, 1983)
- Real Sports Boxing (Atari, 1987)

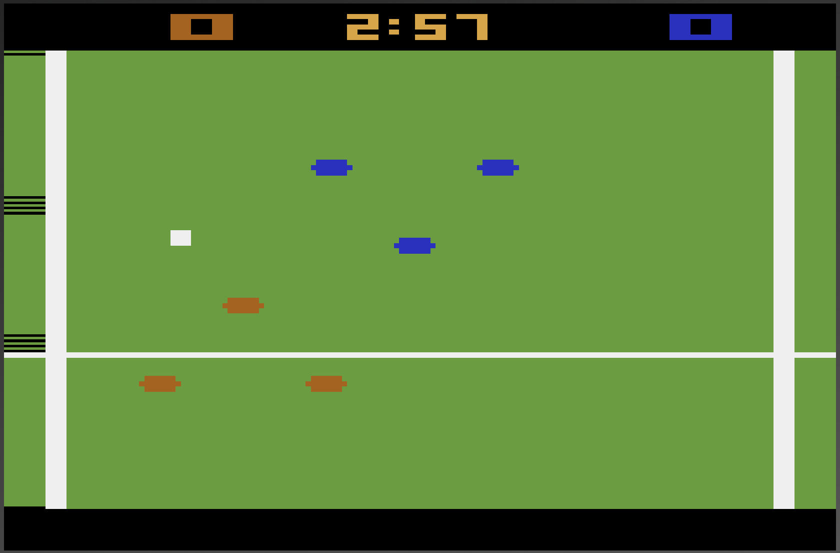
Game impressions (2)



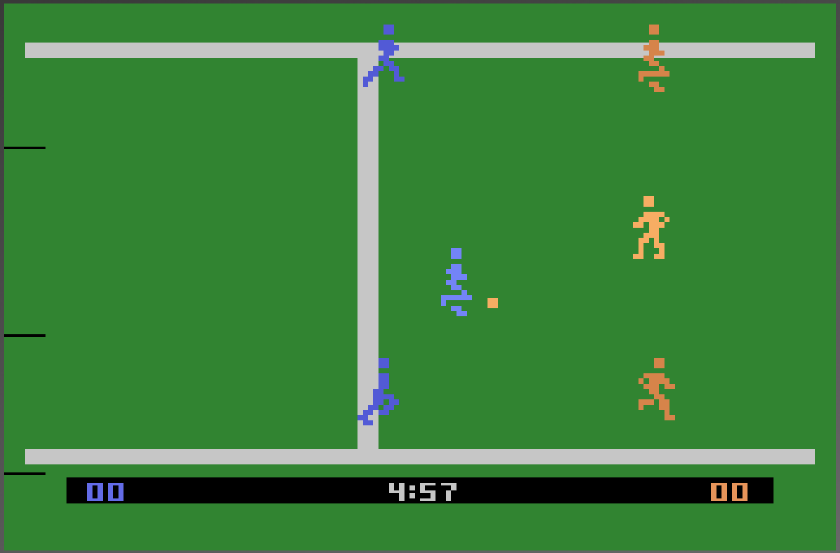
Basketball (Atari, 1978)



Double Dunk (Atari, 1989)



Pelé's Soccer (Atari, 1981)



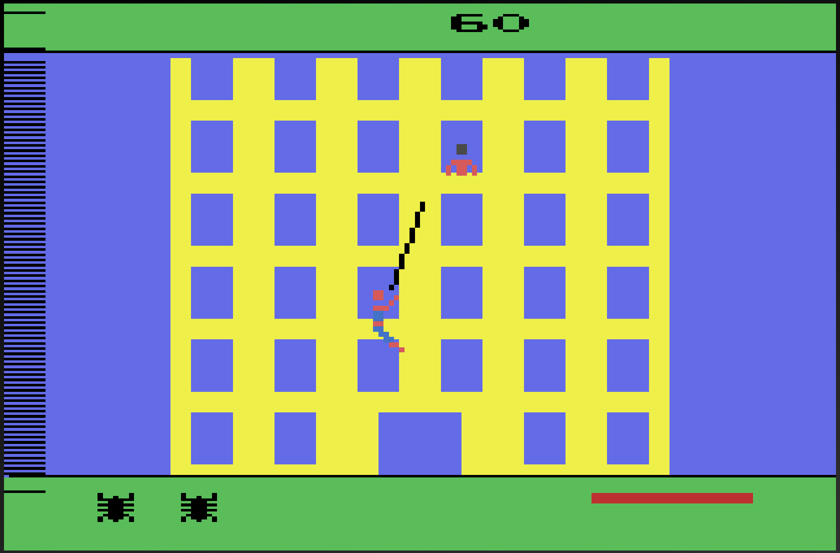
Real Sports Soccer (Atari, 1987)

Inspirations for games (3)

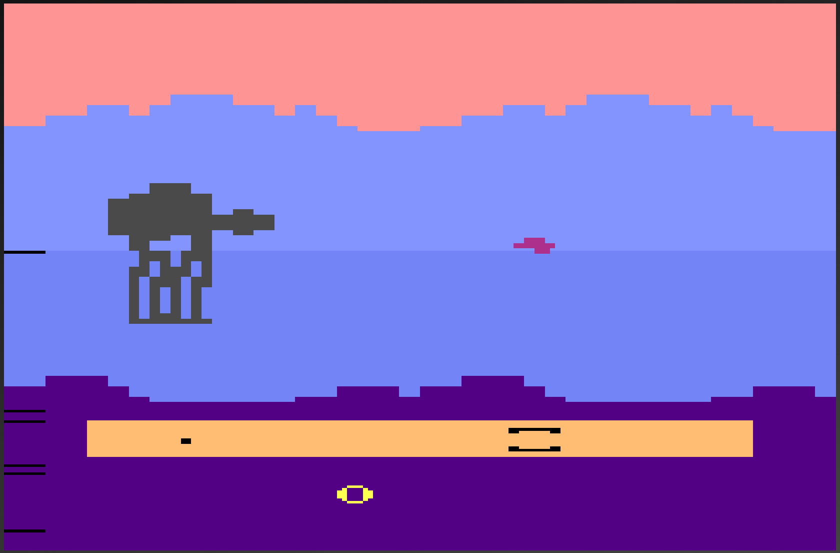
Licensed franchise games

- E.T. (Atari, 1982)
- Indiana Jones: Raiders Of The Lost Ark (Atari, 1982)
- Muppets: Pigs In Space (Atari, 1983)
- Peanuts: Snoopy And The Red Baron (Atari, 1983)
- Smurfs (2 Titles, Coleco, 1982 - 1983)
- Spider-Man (Parker Brothers, 1982)
- Superman (Atari, 1978)
- Star Wars (5 Titles, Parker Brothers, 1982 - 1983)
- Chuck Norris Superkicks (Xonox, 1983)

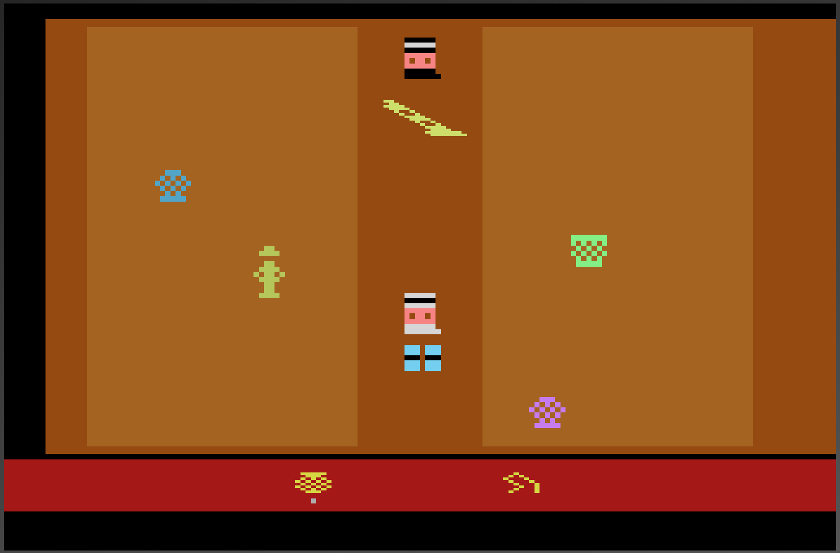
Game impressions (3)



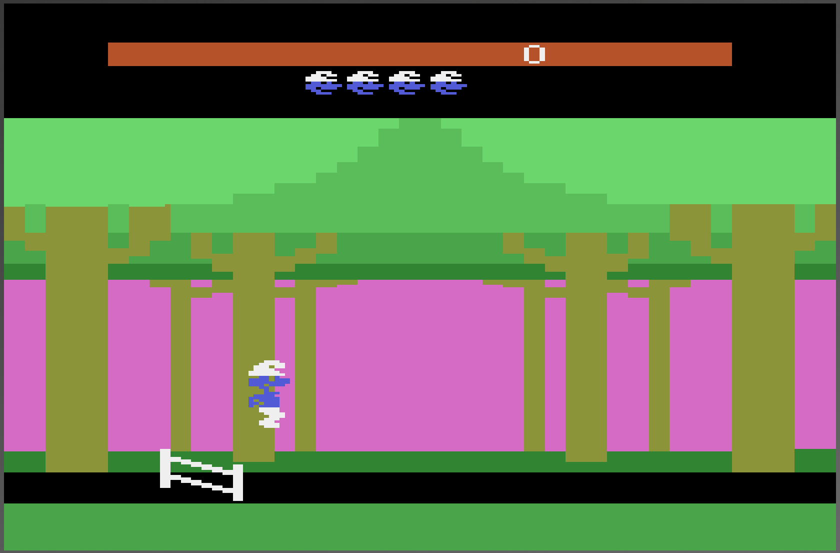
Spider-Man (Parker Bros, 1982)



Empire Strikes Back (Parker Bros, 1982)



Raiders Of The Lost Ark (Atari, 1982)



Smurf's Rescue in ... (CBS, 1982)

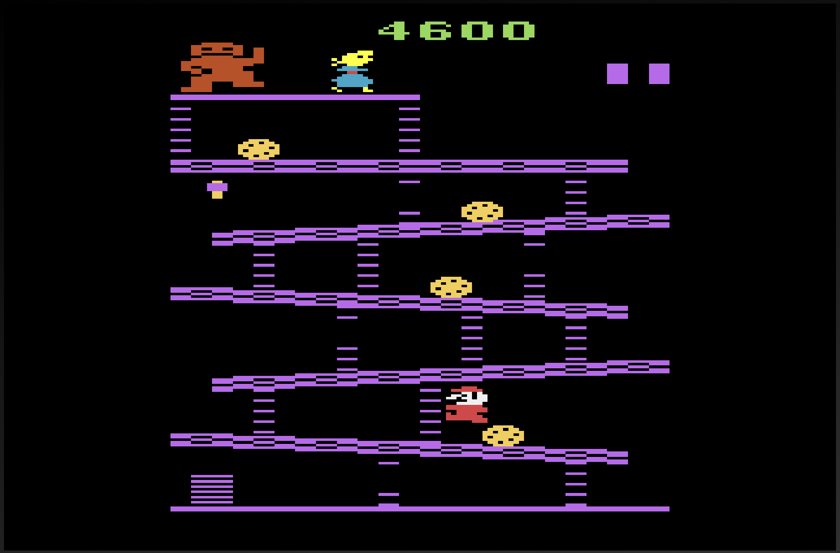
Inspirations for games (4)

Arcade "ports"

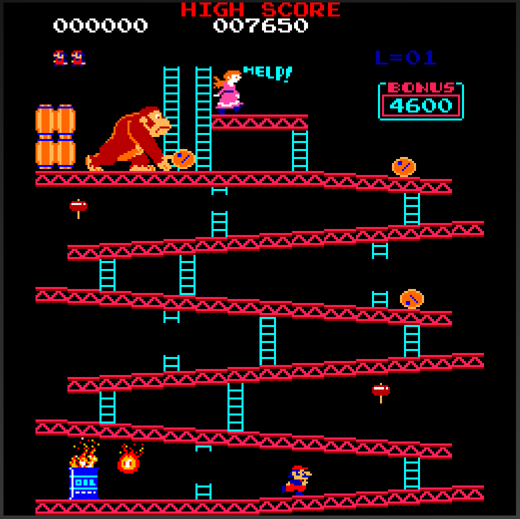
Not really ports, but reimplementations of the game's basic ideas

- Amidar (Parker Bros, 1983)
- Asteroids (Atari, 1981)
- Berzerk (Atari, 1982)
- Breakout (Atari, 1978)
- Combat / Tank (Atari, 1977)
- Defender (Atari, 1981)
- Dig Dug (Atari, 1983)
- Donkey Kong (Coleco, 1982)
- Galaxian (Atari 1983)
- Gyruss (Parker Bros, 1984)
- Kangaroo (Atari, 1983)
- Pac Man (Atari, 1981)
- Pole Position (Atari, 1983)
- Popeye (Parker Bros, 1983)
- Q-Bert (Parker Bros, 1983)
- Phoenix (Atari, 1983)
- Space Invaders (Atari, 1978)
- Zaxxon (CBS, 1982)

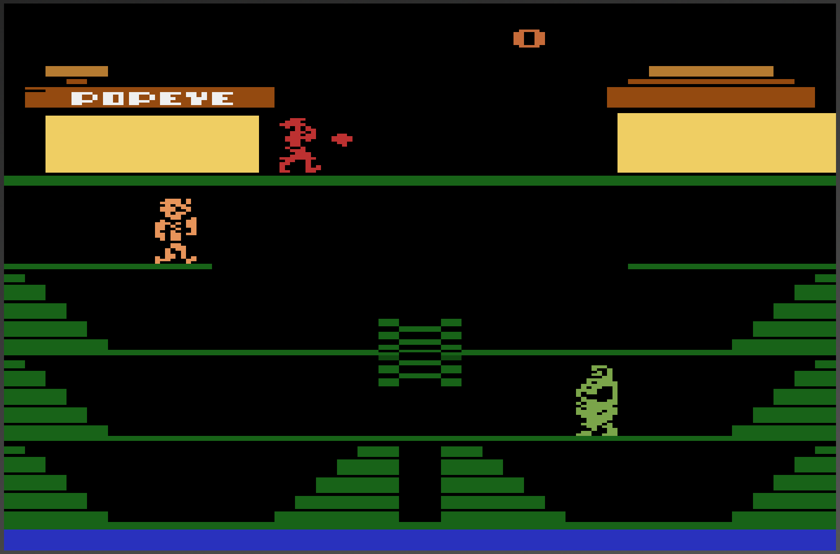
Game impressions (4)



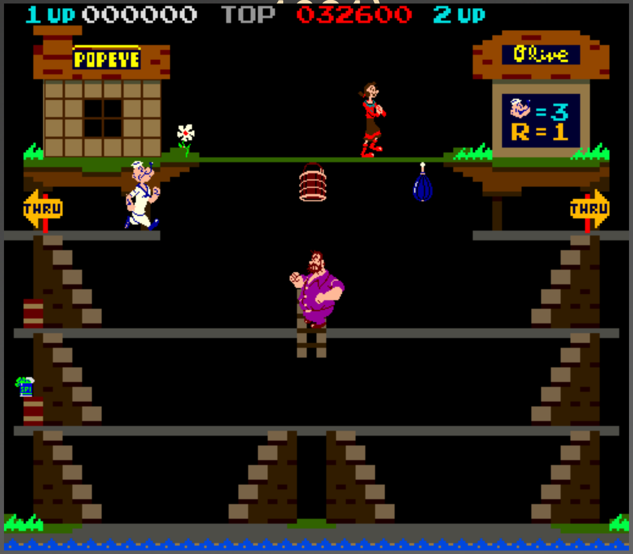
2600: Donkey Kong (Coleco, 1982)



Arcade: Donkey Kong (Nintendo, 1981)



2600: Popeye (Parker Bros, 1983)



Arcade: Popeye (Nintendo, 1982)

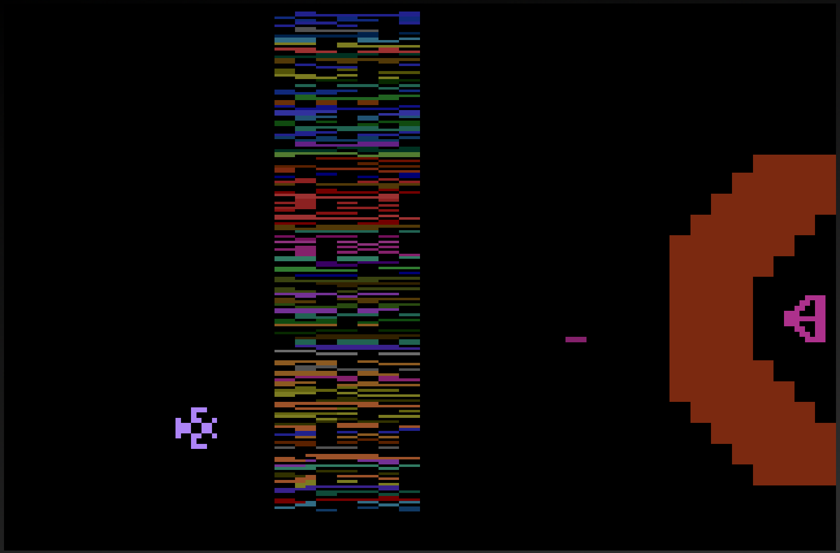
Inspirations for games (5)

Original titles

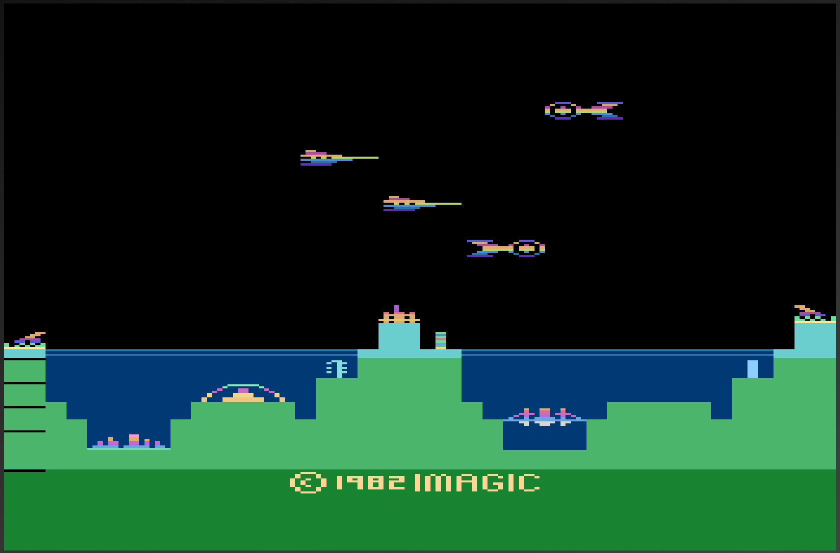
- Adventure (Atari, 1978)
- Atlantis (Imagic, 1982)
- Demon Attack (Imagic, 1982)
- Fathom (Imagic, 1983)
- Haunted House (Atari, 1982)
- H.E.R.O. (Activision, 1984)
- Pitfall! (Activision, 1982)
- Solaris (Atari, 1986)
- Yar's Revenge (Atari, 1982)

Most original titles were
yet another release
for a successful genre

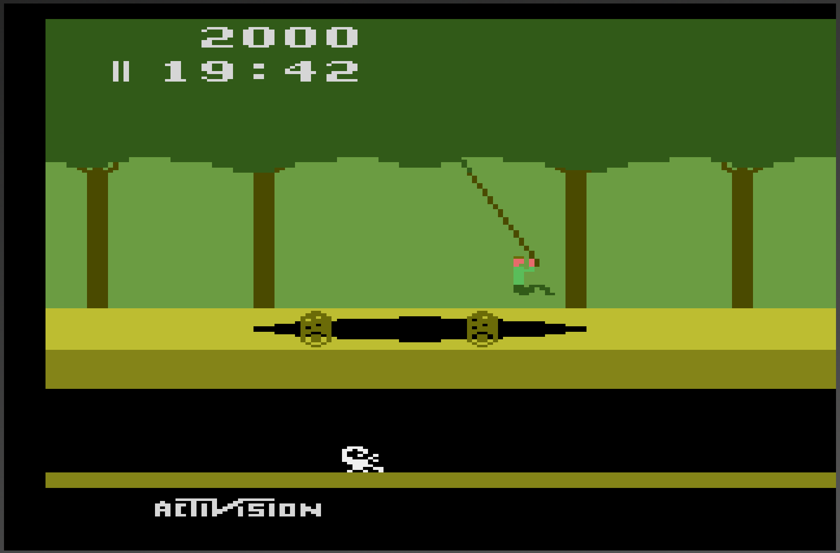
Game impressions (5)



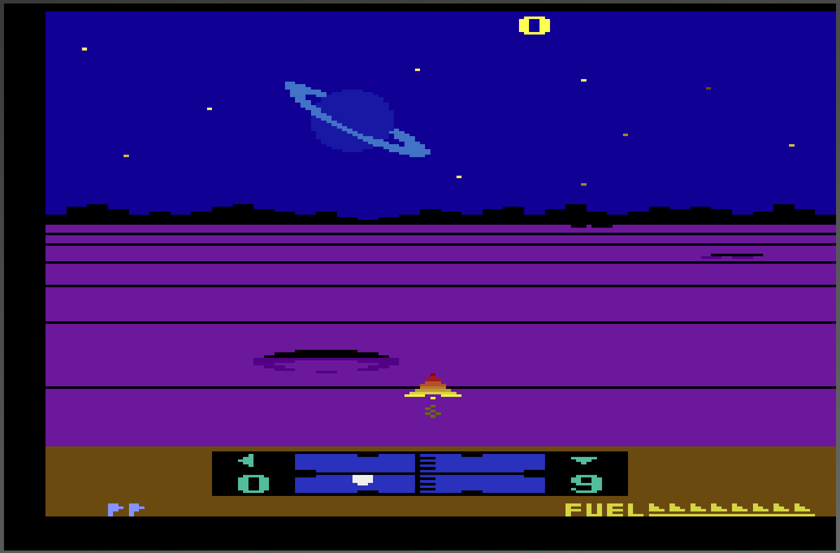
Yar's Revenge (Atari, 1982)



Atlantis (Imagic, 1982)



Pitfall (Activision, 1982)



Solaris (Atari, 1986)

Adventure (1)

Why Adventure as an example?

The ancestor of all action adventures, e.g. Zelda

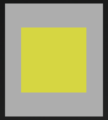
Still a fine game to play

The author Warren Robinett created small website about Adventure: http://www.warrenrobinett.com/adventure/

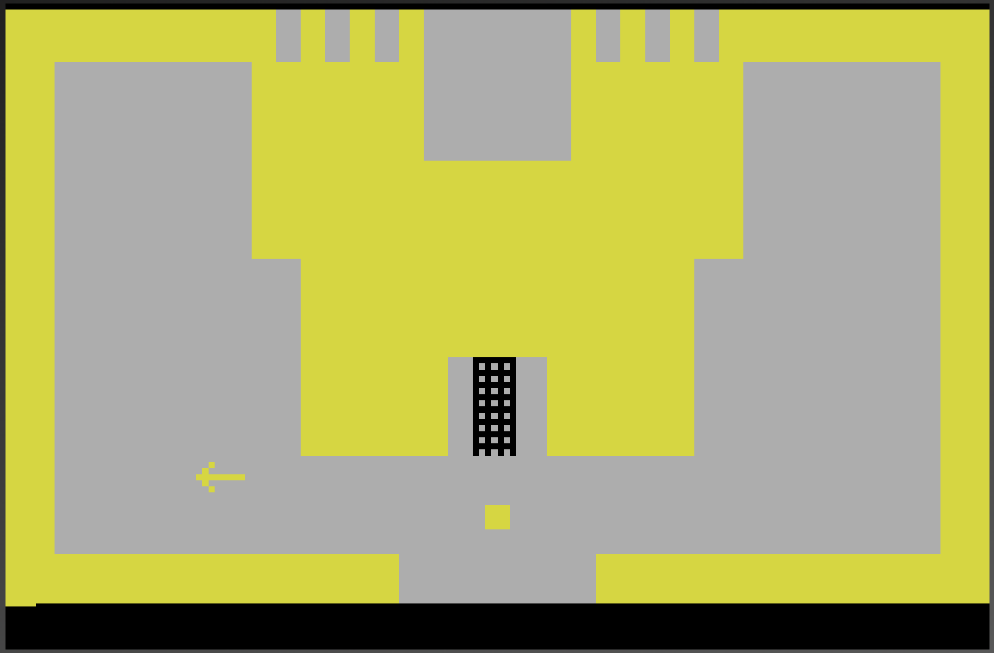
This includes slides for a lecture he gave about the game and Atari 2600 development in general

Adventure (2)

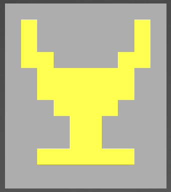
You



are an adventurer and start in front of a yellow castle



Your quest is to bring the enchanted chalice



back to that castle

Adventure (3)

The world of Adventure is divided into 29 screens like the starting screen

3 castles (yellow, white, black)

3 mazes (consisting of several non-linear screens)

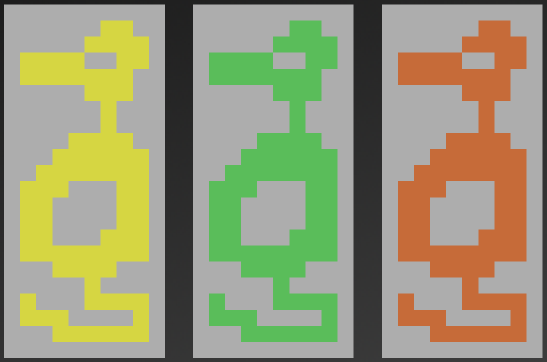
Several "connecting" rooms

Some dead ends that might contain objects

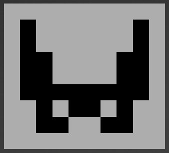
Adventure (4)

Objects interact by overlapping (touching)

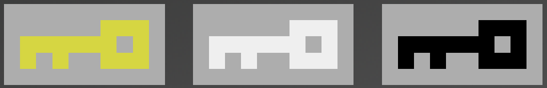
3 dragons to chase you



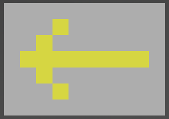
A bat that moves objects on its own



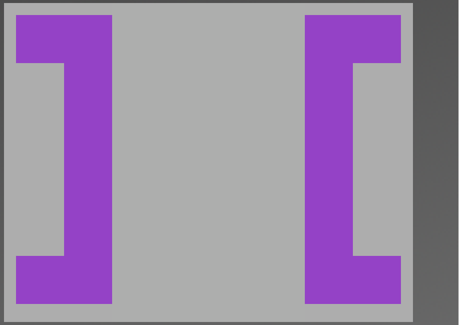
3 keys to the castles



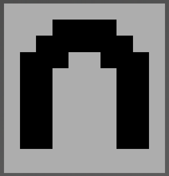
A sword to kill dragons



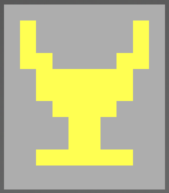
A bridge to cross horizontal walls



A magnet to attract objects



The enchanted chalice



Adventure (5)

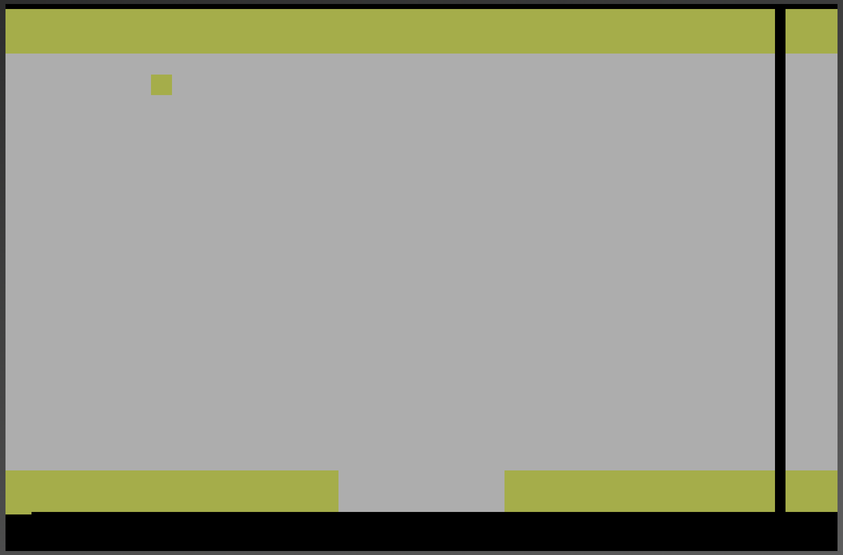
The is one more object

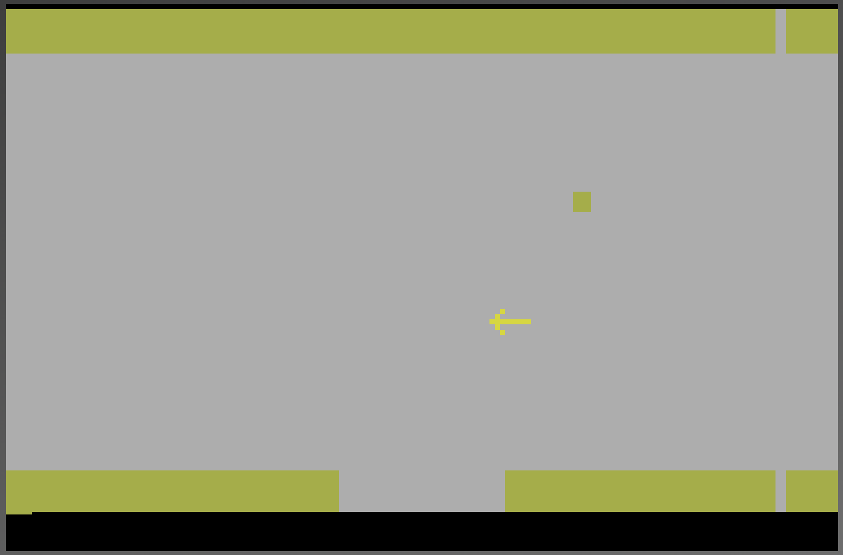
One pixel in size and colored like the background

Hidden in a room only accessible using the bridge

Take this "dot" to a certain room

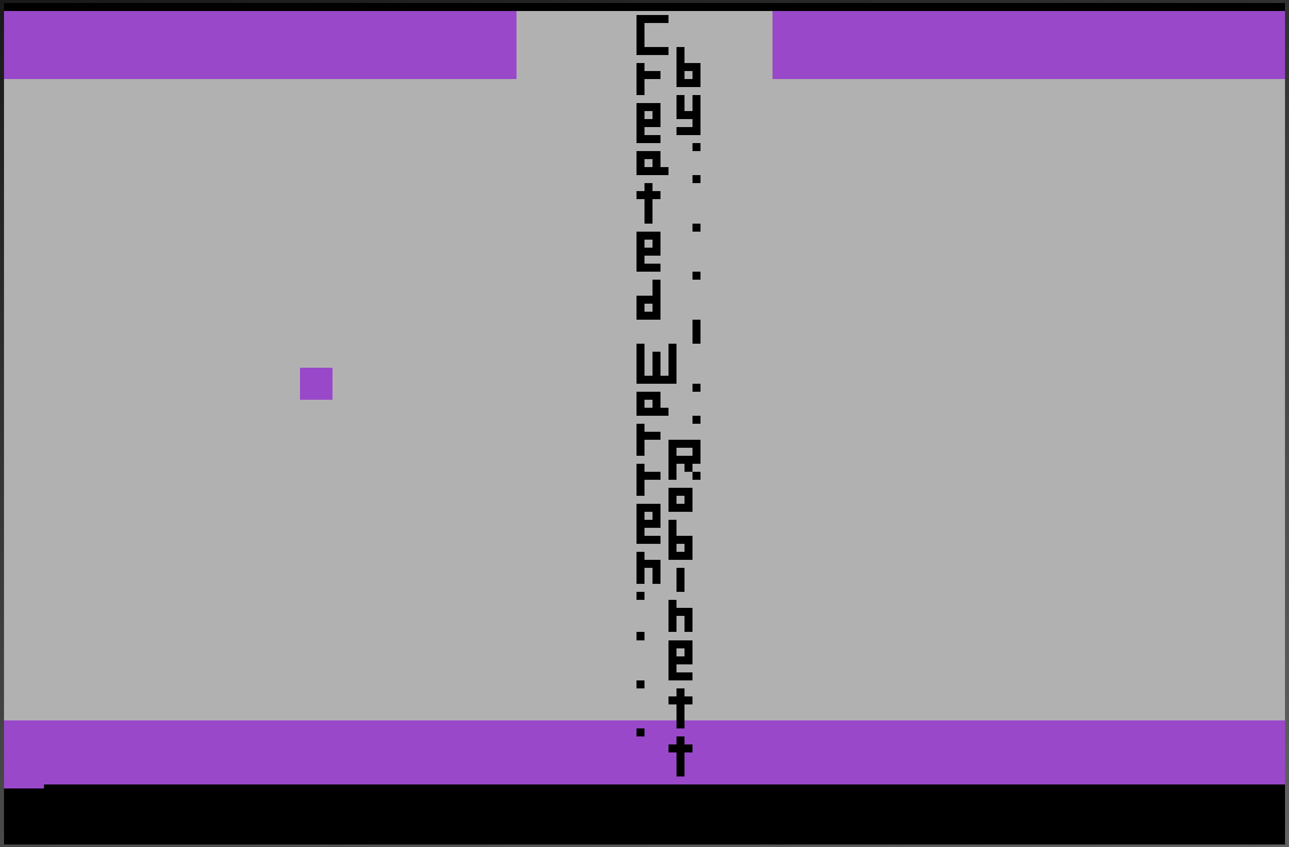
Add another object and the wall is gone





Adventure (6)

Walk though that removed wall and witness the first ever easter-egg of video gaming:



Created by Warren Robinett

The revenge of a disgruntled video game programmer

Development (1)

Programming in 1977:

Code assembled on a computer running a proprietary OS

Connected to a special cartridge

When the software crashed, stripes top down would be displayed

For debugging a logic analyzer was used, which could display steps leading to a special condition

Development (2)

Programming today:

Code assembled on a computer running almost any OS (Windows, Mac OS X, Linux / Unix, ...)

Run inside an emulator with very sophisticated debugging options

Once it works in the emulator as expected, it is transferred to a special cartridge like Supercharger (1983), or Harmony Cart (2009)

Supercharger

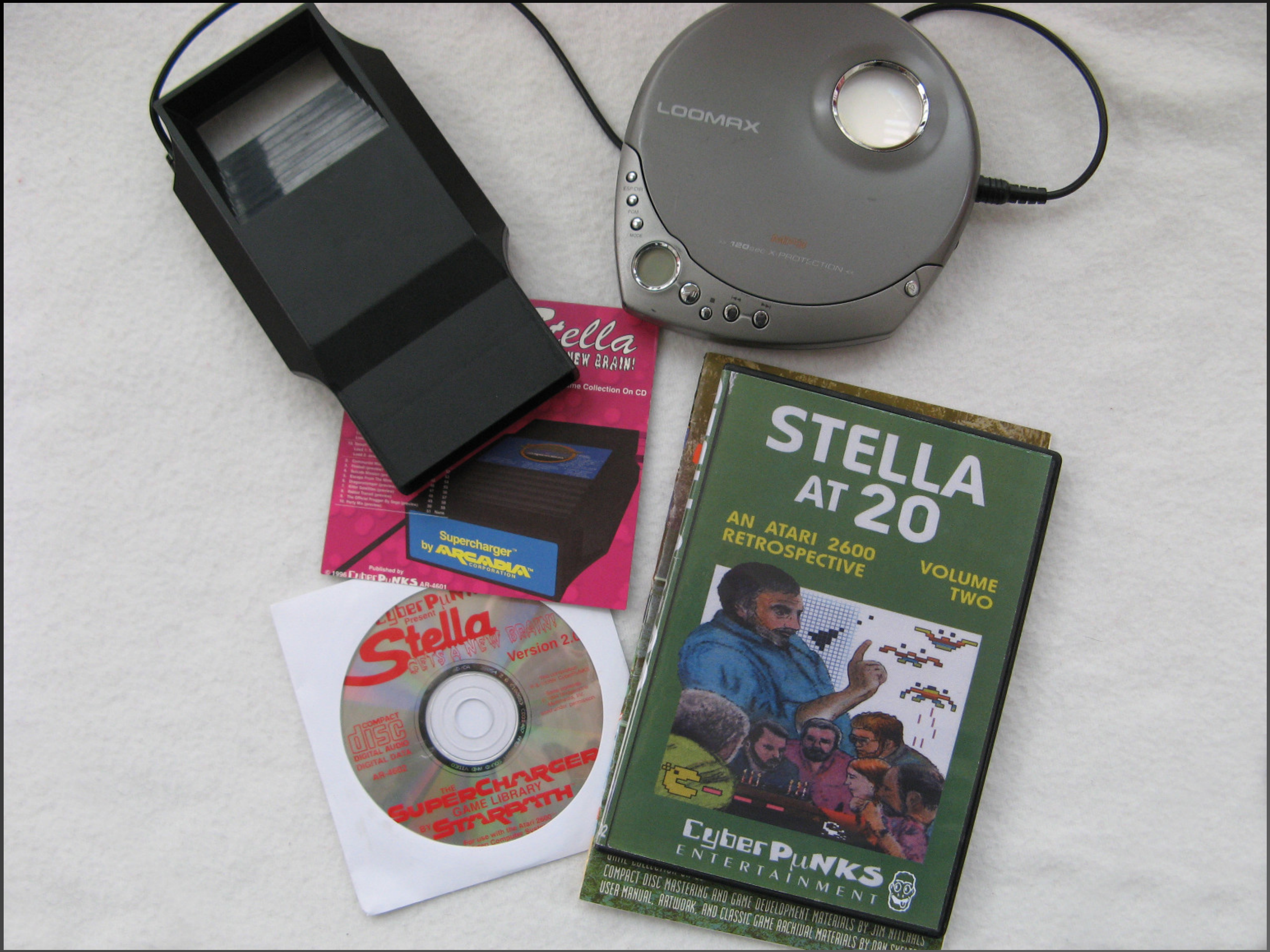
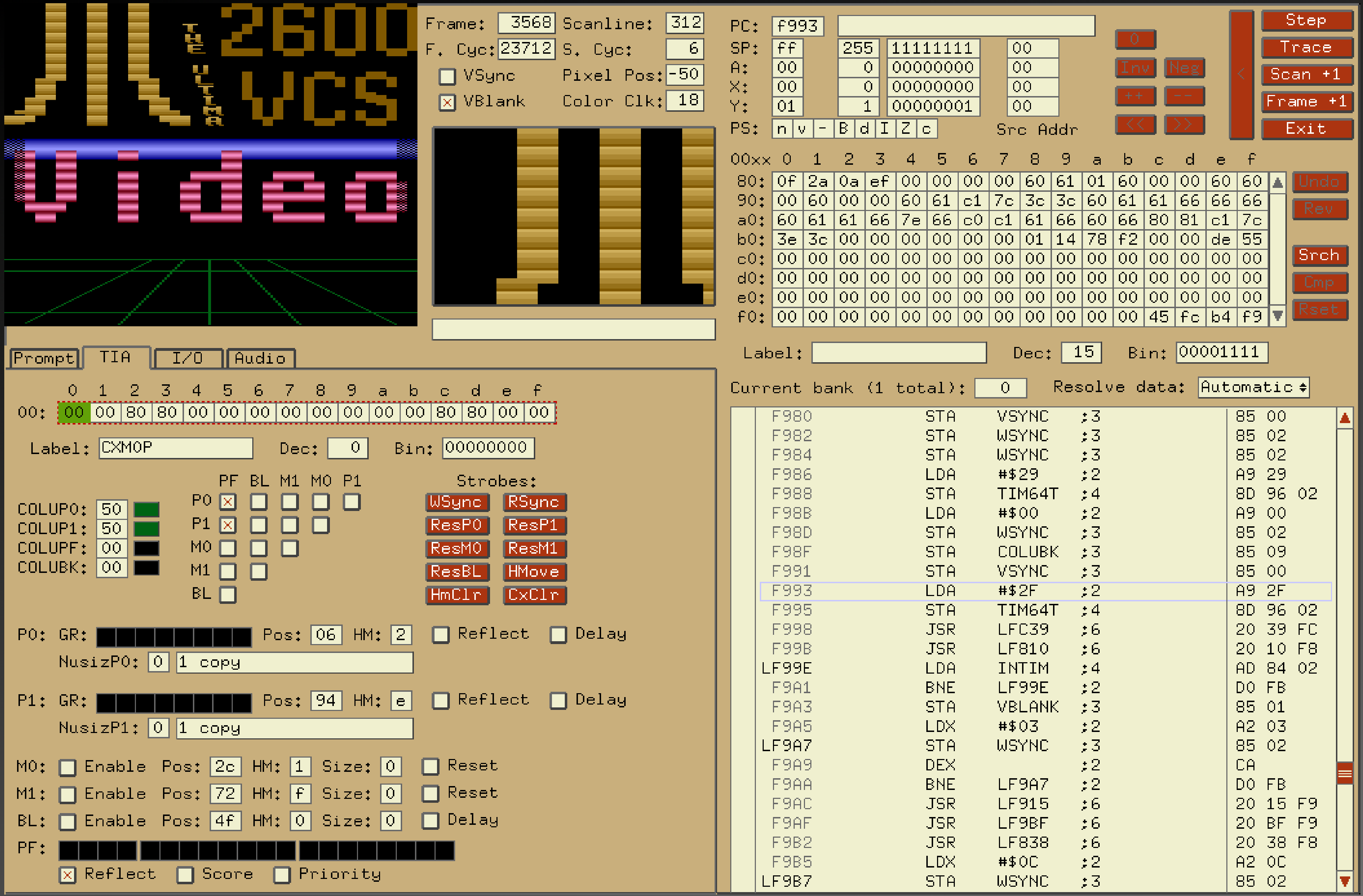

LOOMAX
Stella
New Brain!
Game Collection On CD
Supercharger by ARCADIA CORPORATION
CyberPuNKS
Stella Gets a New Brain!
Version 2.0
The SUPERCHARGER GAME LIBRARY by STARPATH
STELLA AT 20
AN ATARI 2600 RETROSPECTIVE
VOLUME TWO
CyberPuNKS ENTERTAINMENT
COMPACT DISC MASTERING AND GAME DEVELOPMENT MATERIALS BY JIM NITCHALS
USER MANUAL, ARTWORK, AND CLASSIC GAME ARCHIVAL MATERIALS BY DAN SKELTON

Image courtesy of www.ccmuseum.de, used by permission

Stella Debugger



Frame: 3568 Scanline: 312
F. Cyc: 23712 S. Cyc: 6
□ VSync Pixel Pos: -50
☒ VBlank Color Clk: 18

```
PC: f993
SP: ff    255  11111111   00
A:  00      0  00000000   00
X:  00      0  00000000   00
Y:  01      1  00000001   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: 0f 2a 0a ef 00 00 00 00 60 61 01 60 00 00 60 60
 90: 00 60 00 00 60 61 c1 7c 3c 3c 60 61 61 66 66 66
 a0: 60 61 61 66 7e 66 c0 c1 61 66 60 66 80 81 c1 7c
 b0: 3e 3c 00 00 00 00 00 00 01 14 78 f2 00 00 de 55
 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 45 fc b4 f9
```

Label: Dec: 15 Bin: 00001111

Current bank (1 total): 0 Resolve data: Automatic

```
 F980        STA   VSYNC   ;3     85 00
 F982        STA   WSYNC   ;3     85 02
 F984        STA   WSYNC   ;3     85 02
 F986        LDA   #$29    ;2     A9 29
 F988        STA   TIM64T  ;4     8D 96 02
 F98B        LDA   #$00    ;2     A9 00
 F98D        STA   WSYNC   ;3     85 02
 F98F        STA   COLUBK  ;3     85 09
 F991        STA   VSYNC   ;3     85 00
 F993        LDA   #$2F    ;2     A9 2F
 F995        STA   TIM64T  ;4     8D 96 02
 F998        JSR   LFC39   ;6     20 39 FC
 F99B        JSR   LF810   ;6     20 10 F8
LF99E        LDA   INTIM   ;4     AD 84 02
 F9A1        BNE   LF99E   ;2     D0 FB
 F9A3        STA   VBLANK  ;3     85 01
 F9A5        LDX   #$03    ;2     A2 03
LF9A7        STA   WSYNC   ;3     85 02
 F9A9        DEX           ;2     CA
 F9AA        BNE   LF9A7   ;2     D0 FB
 F9AC        JSR   LF915   ;6     20 15 F9
 F9AF        JSR   LF9BF   ;6     20 BF F9
 F9B2        JSR   LF838   ;6     20 38 F8
 F9B5        LDX   #$0C    ;2     A2 0C
LF9B7        STA   WSYNC   ;3     85 02
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 00 00 80 80 00 00 00 00 00 00 00 00 80 80 00 00
```

Label: CXM0P Dec: 0 Bin: 00000000

	PF	BL	M1	M0	P1
P0	☒	□	□	□	□
P1	☒	□	□	□	
M0	□	□	□		
M1	□	□			
BL	□				

Strobes:

COLUP0: 50 🟩
COLUP1: 50 🟩
COLUPF: 00 ■
COLUBK: 00 ■

P0: GR: ■■■■■■■■ Pos: 06 HM: 2 □ Reflect □ Delay
NusizP0: 0 1 copy

P1: GR: ■■■■■■■■ Pos: 94 HM: e □ Reflect □ Delay
NusizP1: 0 1 copy

M0: □ Enable Pos: 2c HM: 1 Size: 0 □ Reset
M1: □ Enable Pos: 72 HM: f Size: 0 □ Reset
BL: □ Enable Pos: 4f HM: 0 Size: 0 □ Delay
PF: ■■■■ ■■■■■■■■ ■■■■■■■■
☒ Reflect □ Score □ Priority

Piracy (1)

In the design phase copy protection was no issue

The hardware was too sophisticated

Suddenly there was competition: Activision

Activision was founded by four Atari developers who were told: "You are no more important than the guy who puts the cartridge in the box."

Atari filed a lawsuit to prohibit third party game development and lost

Other companies like Tigervision, Parker Bros, and Imagic started unauthorized game development

Piracy (2)

In the early 80's ordinary piracy became an issue

Why code a new game, when you just can replace the company's logo?

There was even an option to copy games at home:

Unimex Duplicator SP 280

Atari filed a lawsuit to prohibit distribution and won

Homebrew (1)

Even with the 2600 being out of production for decades, new titles are released every year

Developed by a homebrew community

AtariAge and others sell cartridges of these releases

100+ titles have been released

A lot of the homebrew games outperform the "original" software from 1977 - 1991

Homebrew (2)

Two different assemblers can be recommended:

DASM: de-facto standard of the 2600 homebrew community

CC65: full featured cross device 6502 tool chain, including C compiler (subset), assembler and linker targeting Apple, Atari and Commodore and other 8 bit computers

For an easy introduction there is batari Basic
→ BASIC-like language that compiles to assembler source code for DASM

Homebrew (3)

You don't want to write a new game from scratch?

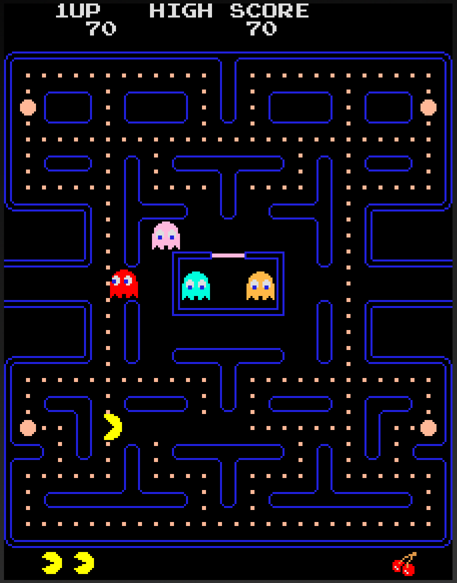
Go hacking and modding other games

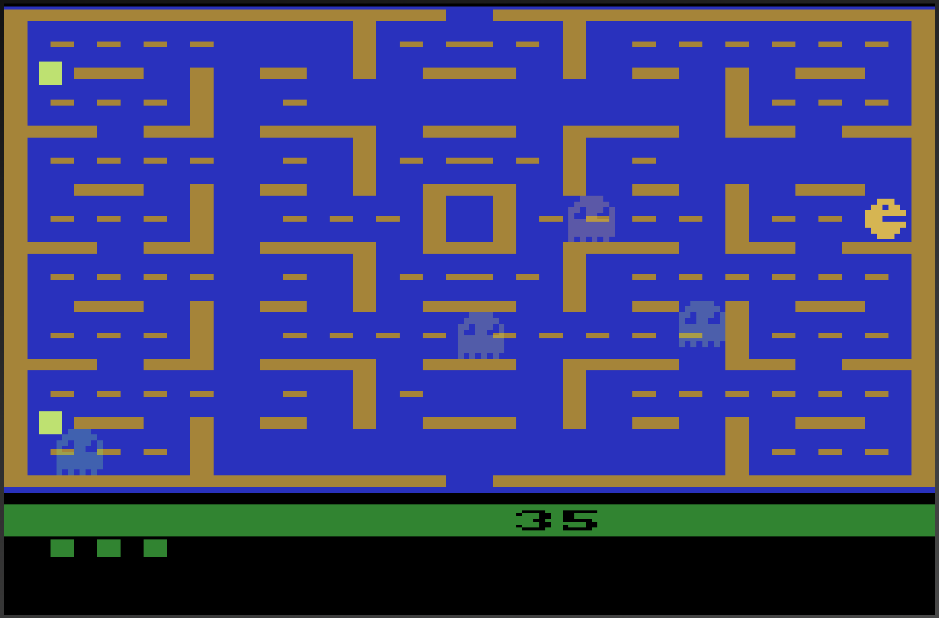
- Just change the graphics
- For a few games there are even editors (Combat, Adventure)
- Disassemble a game and modify it

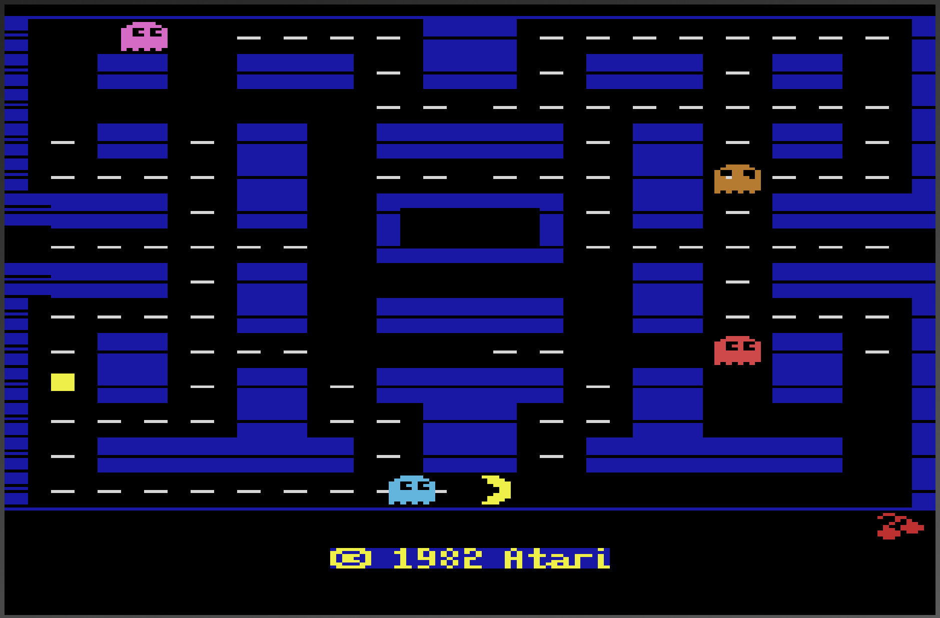
Homebrew (4)

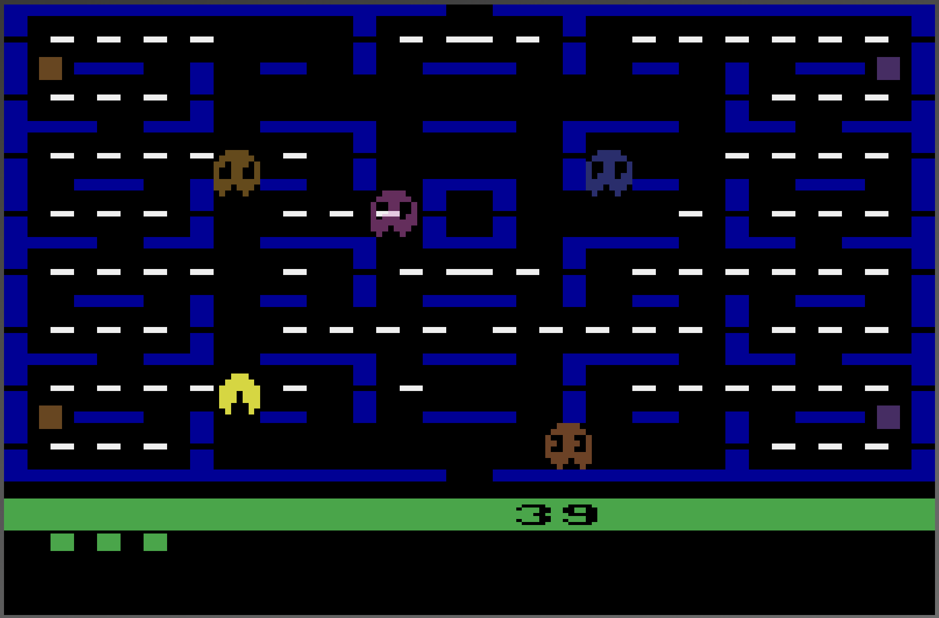
Example for hacking game ROMs:

Pac Man





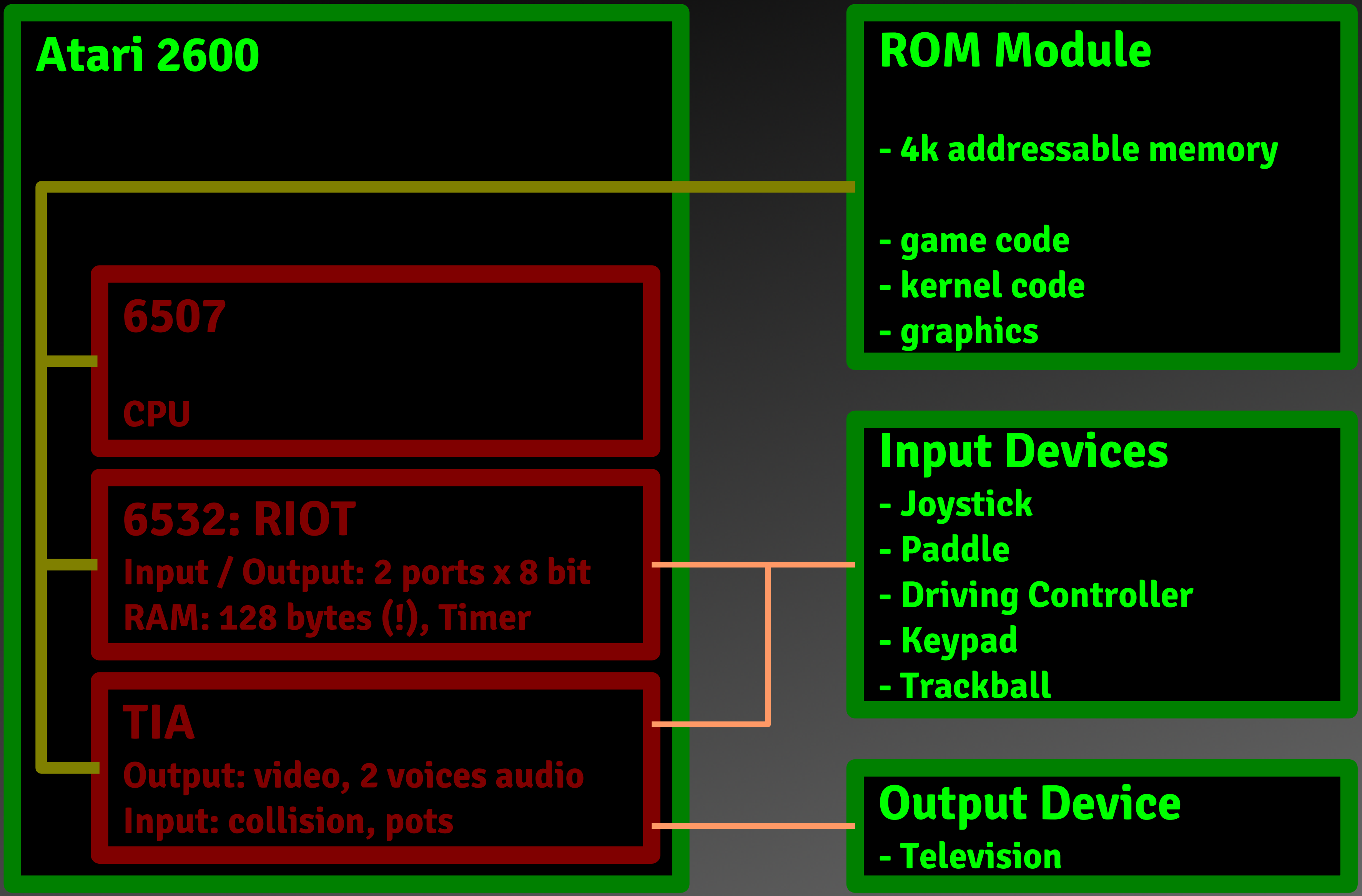




Part 2:

The hardware

Hardware block diagram



Atari 2600

6507

CPU

6532: RIOT

Input / Output: 2 ports x 8 bit

RAM: 128 bytes (!), Timer

TIA

Output: video, 2 voices audio

Input: collision, pots

ROM Module

- 4k addressable memory

- game code
- kernel code
- graphics

Input Devices
- Joystick
- Paddle
- Driving Controller
- Keypad
- Trackball

Output Device
- Television

6507: the CPU (1)

The 6507 is a stripped down version of the 6502

Described in depth by Michael Steil on 27c3

Here's only a very brief overview of the 6502

Designed by Chuck Peddle, who also worked on the Motorola 6800 team

8 bit architecture, little endian

Instructions take 1 - 3 bytes and 2 - 7 clock cycles

Clocked at ~1.19MHz

Cheap in production, competitive in speed

6507: the CPU (2)

6 registers

A: multi-purpose accumulator (8 bit)

X: index register (8 bit)
Y: index register (8 bit)

PC: program counter (16 bit)

SP: stack pointer (8 bit) (offset to $0100)

ST: processor status (8 bit)

6507: the CPU (3)

Let's compare the 6507 to the 6502:

Smaller chip package (28 pins instead of 40 pins)

What's missing?

3 address lines (64k internal, but only 8k external)

Both interrupt lines are hardwired to +5V internally

1 clock line (phi1), 1 VSS, Sync, S0

3 "n.c." pins ;-)

Even cheaper, popular for embedded applications

6532: RAM, I/O and Timer

Very common companion chip to the 6502 family

128 bytes of RAM

2 I/O ports (8 bit)

- 1 I/O port used for the 5 console switches
- 1 I/O port used for both joysticks (only directions, read-write)

Timer that is optionally capable of sending interrupts

(6507 is not capable of receiving interrupts, though)

Memory map (1): overview

External address space of 6507 is 8k

Mirrored 8 times in 64k internal address space

Starting at:

- $0000, $2000, $4000, $6000, $8000, $A000, $C000, $E000

$0000 - $0FFF IO, timer and RAM

$1000 - $1FFF ROM (module)

Typically used in two ways:

$0000 - $1FFF
$0000 - $0FFF and $F000 - $FFFF

Memory map (2): TIA

Exact mapping: xxx0 xxxx 0xNN NNNN

Usually accessed at $0000 - $003F

Available at 32 different positions inside 8k area:

- $0000, $0040, $0100, $0140, ..., $0F00, $0F40

"Space" for 64 registers

14 "read only" registers

- Mirrored 4 times inside the 64 bytes address space

45 "write only" registers

Memory map (3): RIOT (1)

Exact mapping: xxx0 xxMx 1NNN NNNN

M: mode (0: RAM 1: I/O+Timer)

RAM: usually accessed at $0080 - $00FF

IO and TIMER: usually accessed at $0280 - $029F

Available 8 times in 8k space, alternating

RAM: $0080, $0180, $0480, $0580, ..., $0C80, $0D80

IO: $0280, $0380, $0680, $0780, ..., $0E80, $0F80

IO-Ports: $0280 - $0283

Timer: $0284 - $028C, $0294 - $0297, $029C - $029F

Memory map (4): RIOT (2)

RAM: 128 bytes

Needed at two locations

- $0080 - $00FF: "variables"
- $0180 - $01FF: stack

Keep in mind that the stack uses a mirror

Quote from development manual:

"The microprocessor stack is normally located from FF on down, and variables are normally located from 80 on up (**hoping the two never meet**)."

Memory map (5): ROM

Cartridge port has 24 connectors

Resembling 24 pins of an 32k bit ROM / EPROM

Power: 3 lines: 1x +5V VCC, 2x GND

D0-D7: 8 data lines

A0-A12: 13 address lines

What's missing?

– Chip select: per definition CS is high active A12

– Read / Write: only defined as ROM port (design fail)

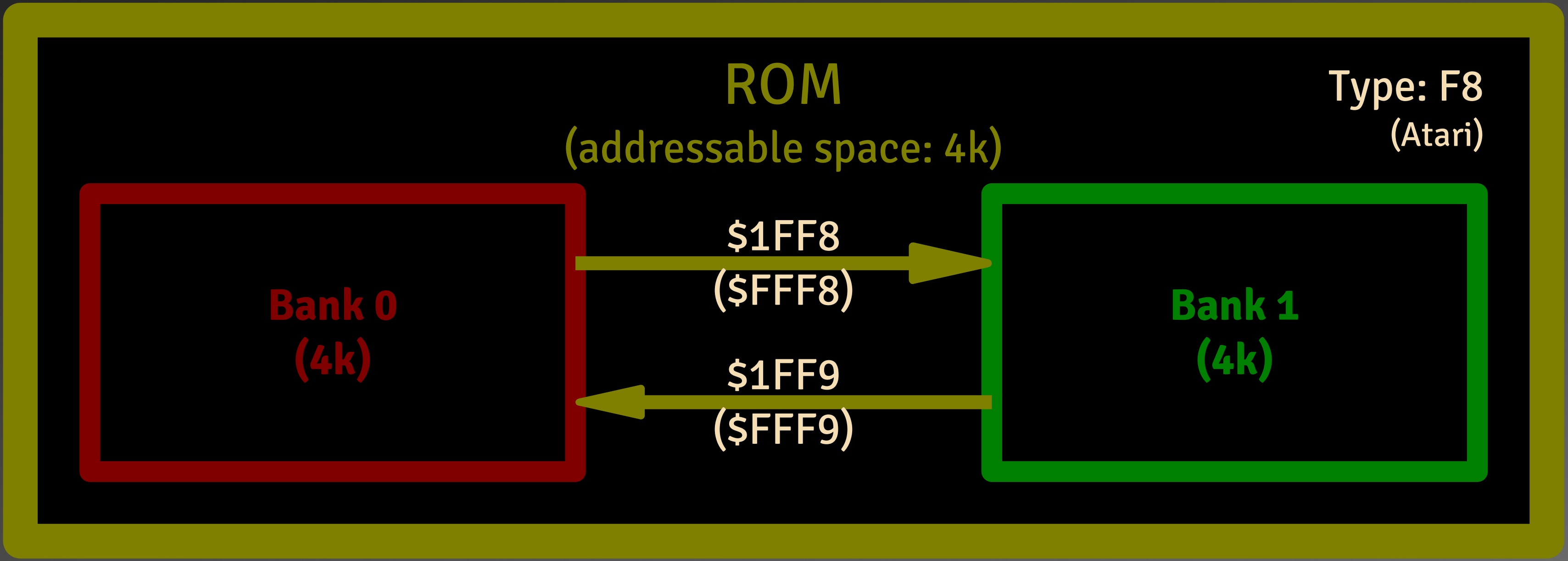
Working around the barriers (1)

At the start (1977) only 2k or 4k ROM modules

At 1981 first 8k ROM modules available

How to fit 8k in a 4k address space?

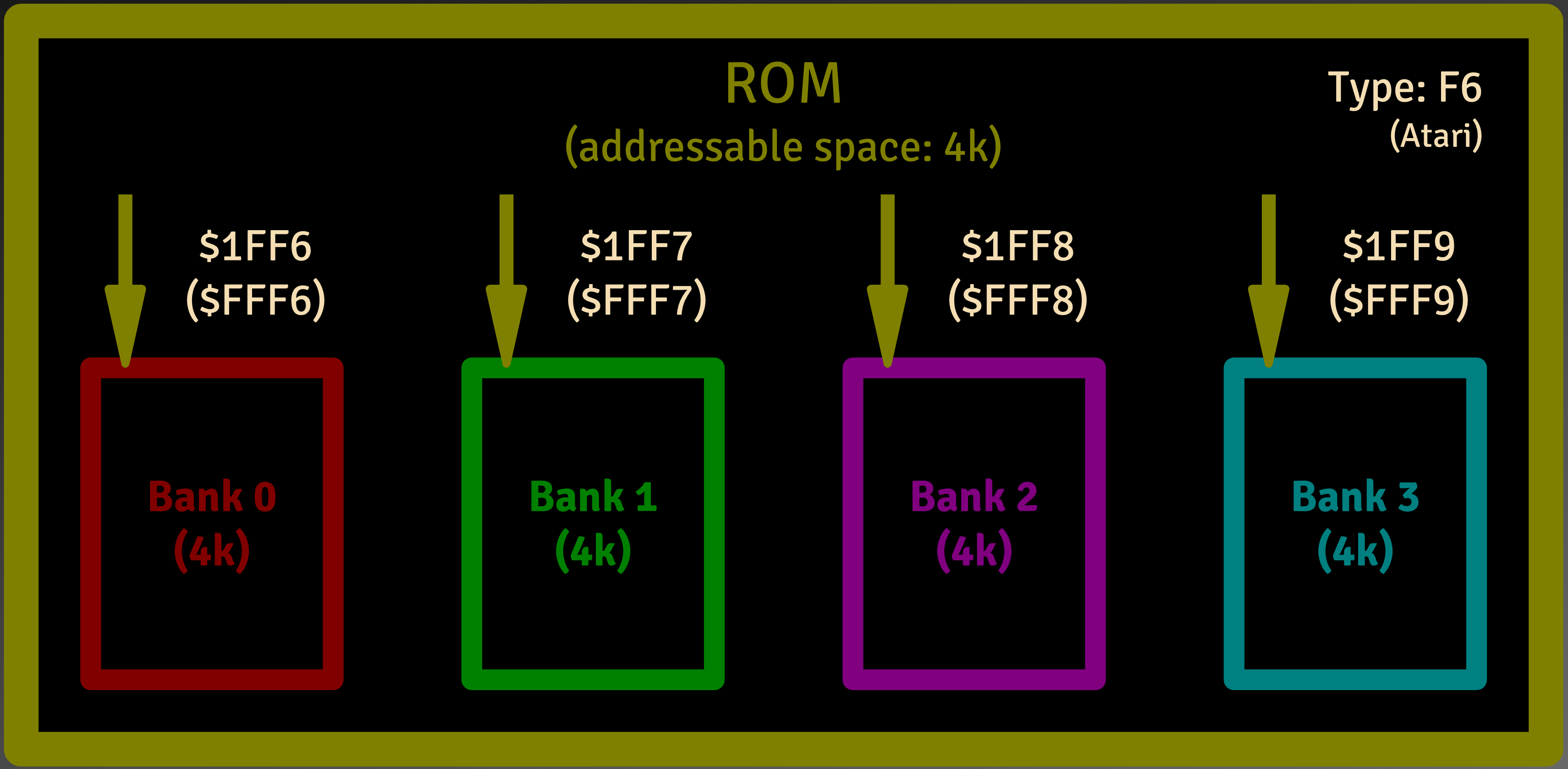
Bank switching!



Bankswitching 16k

What if you need more ROM?

Simple: add more banks!



Working around the barriers (2)

Now that there's enough ROM, how do we get more RAM?

Remember:
no read / write line available on game module

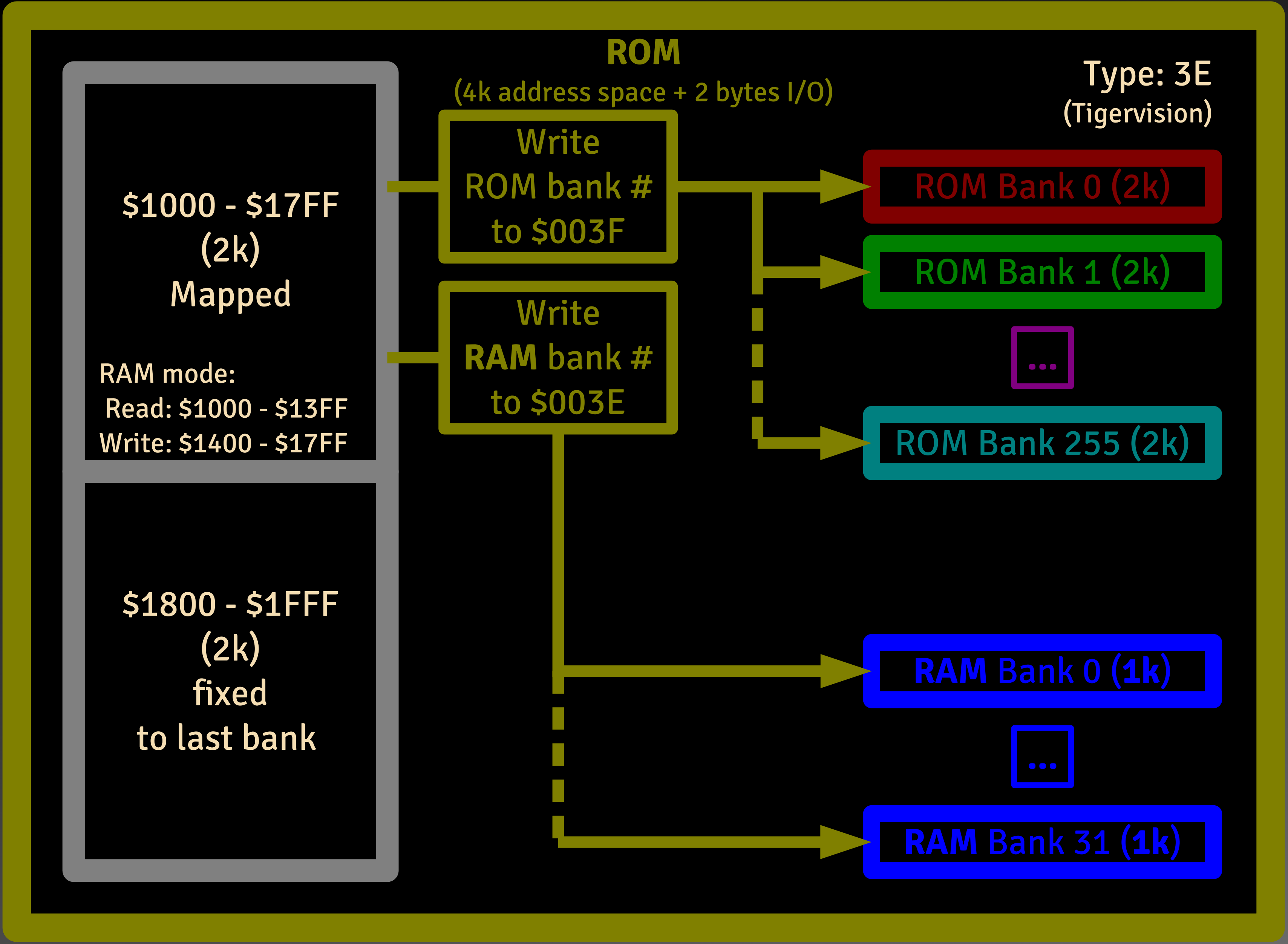
Solution: use different addresses

Write-port:	$1000 - $107F
Read-port:	$1080 - $10FF

Read $1080 to get value written to $1000

Variation of F8: F8SC, and F6: F6SC (Atari)

"Mega-Cartridge"



Working around the barriers (3)

Conclusion:

There are many ways to get more ROM and even RAM into a cartridge

All include some kind of bank-switching scheme

5 real-life cartridge configurations (F8,F6,F8SC,F6SC,3E) have been introduced

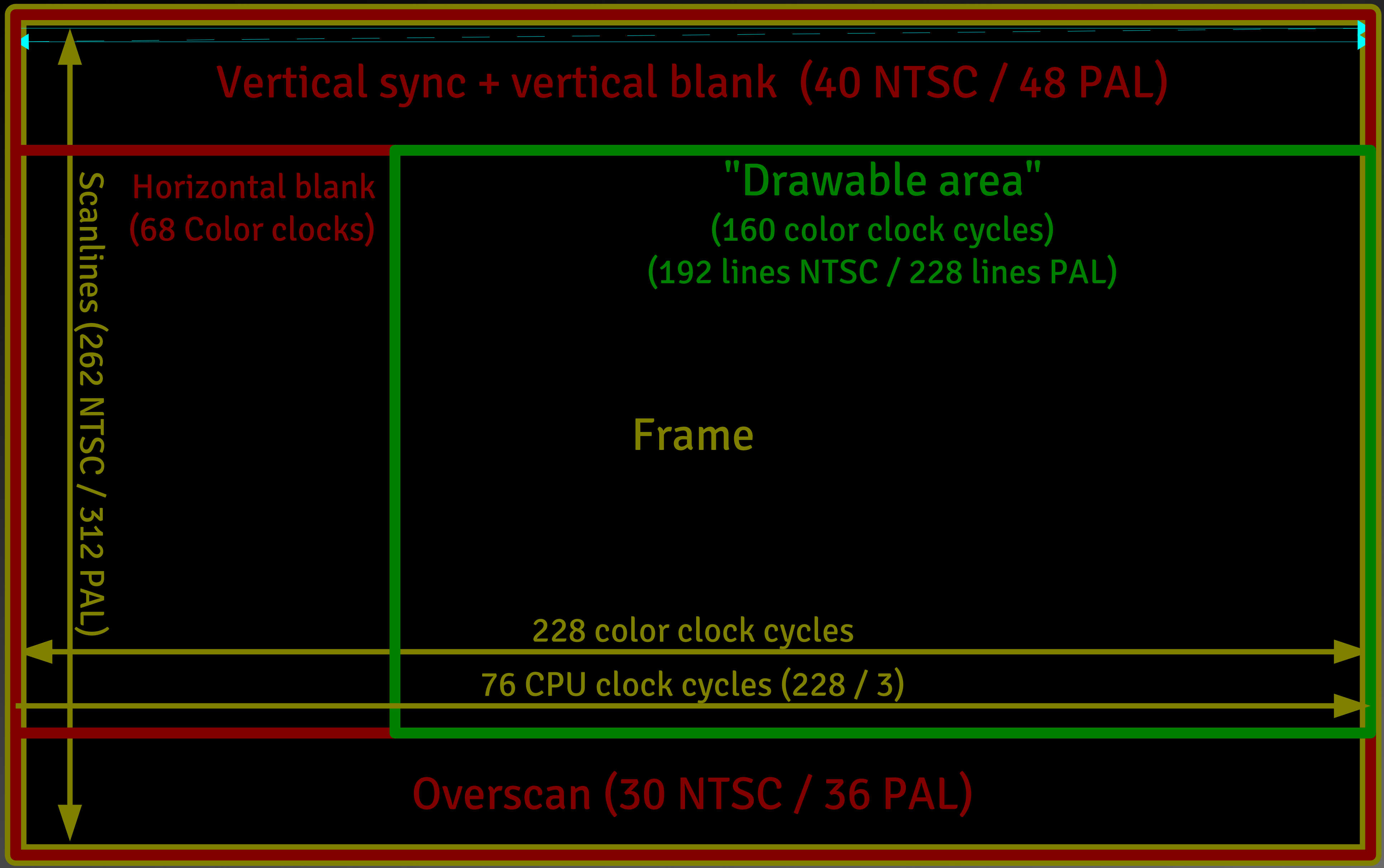
Stella knows 25 different

Defining new schemes is easy nowadays using micro controllers in cartridges

Part 3:

How to write programs

Display



Vertical sync + vertical blank (40 NTSC / 48 PAL)

Horizontal blank
(68 Color clocks)

"Drawable area"
(160 color clock cycles)
(192 lines NTSC / 228 lines PAL)

Scanlines (262 NTSC / 312 PAL)

Frame

228 color clock cycles

76 CPU clock cycles (228 / 3)

Overscan (30 NTSC / 36 PAL)

No Framebuffer

When the Atari 2600 was designed in 1975, RAM was very expensive

To convert the graphics capabilities to a dumb framebuffer you'll need about 30k of 7-bit words

Not only too expensive, but also not addressable by 6507 (8k)

A completely different approach: program the video chip while the image is displayed

Advantage: cheap and very flexible

Disadvantage: CPU is "occupied" during display

"Racing the beam" (1)

Instead of "running" the graphics frame by frame, the image is drawn line by line

If nothing is changed, the next line is drawn like the one before

There are no registers for Y-components

Example: sprite size is 8 bit wide and as high as the screen

You need to tell the TIA what to paint while it is painting! This is called "Racing the beam"

"Racing the beam" (2)

Write registers of the TIA:

VSYNC	VBLANK	WSYNC	RSYNC	NUSIZ0	NUSIZ1
COLUP0	COLUP1	COLPF	COLBK	CTRLPF	REFP0
REFP1	PF0	PF1	PF2	RESP0	RESP1
RESM0	RESM1	RESBL	AUDC0	AUDC1	AUDF0
AUDF1	AUDV0	AUDV1	GRP0	GRP1	ENAM0
ENAM1	ENABL	HMP0	HMP1	HMM0	HMM1
HMBL	VDELP0	VDELP1	VDELBL	RESMP0	RESMP1
HMOVE	HMCLR	CXCLR		Sync	Graphics

4 registers for syncing, 34 for graphics display

Graphics capabilities

Background color

2 player sprites (8 bit), each with its own color

Playfield with own color
- – can also re-use player colors

2 missile sprites (1 bit), re-using player colors

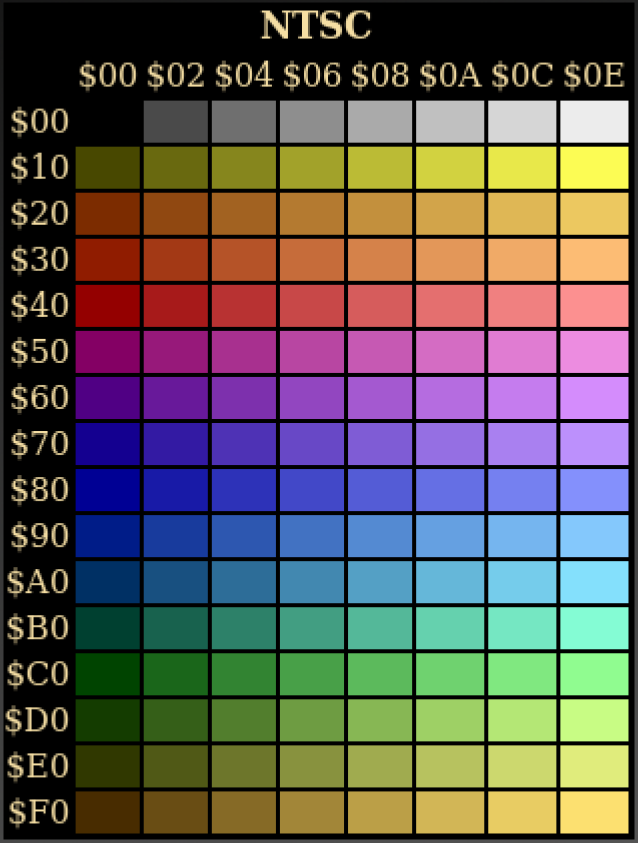
1 ball sprite (1 bit), re-using playfield color

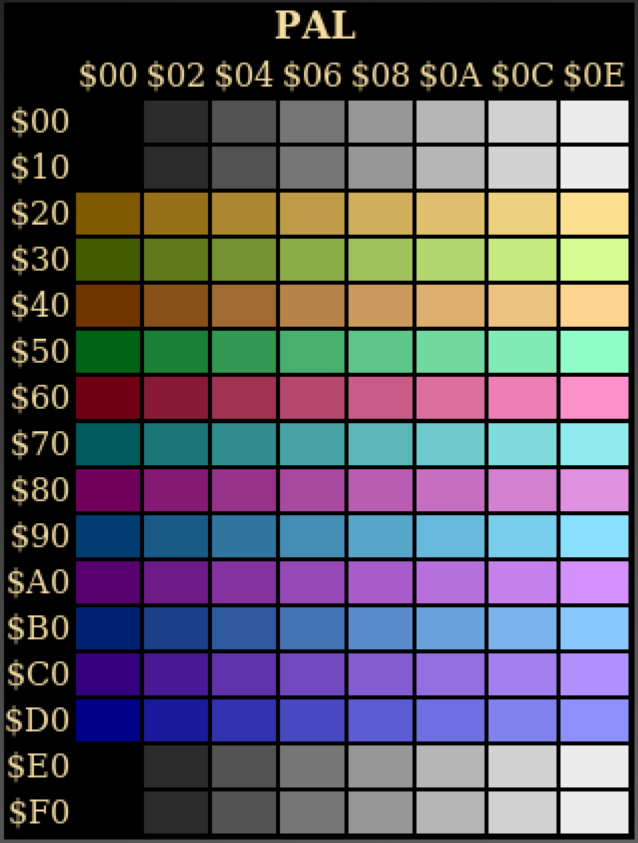
Requirements: run "Combat" and "Pong"

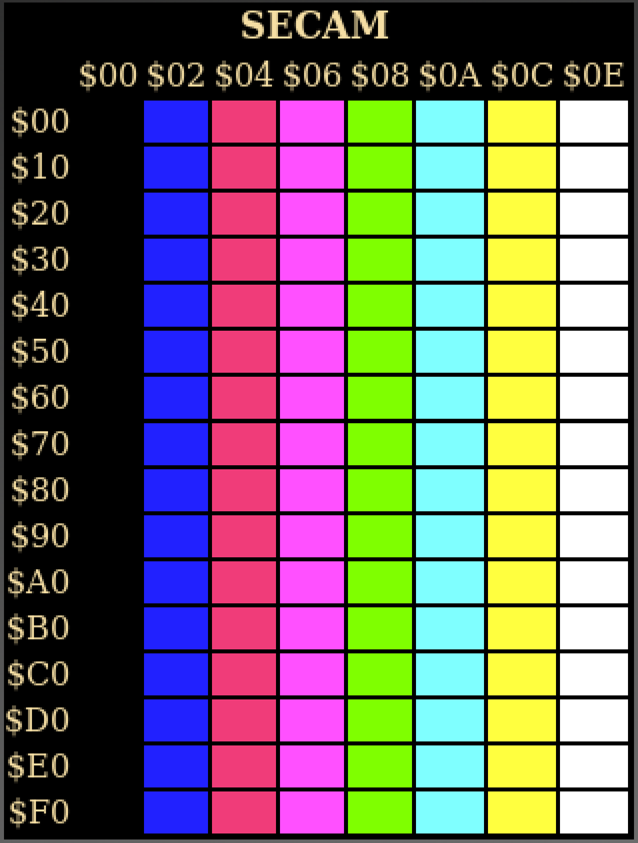
Colors

4 Color registers: background, playfield, 2 players

Each color can be picked out of a palette of 128


NTSC
$00 $02 $04 $06 $08 $0A $0C $0E
$00
$10
$20
$30
$40
$50
$60
$70
$80
$90
$A0
$B0
$C0
$D0
$E0
$F0


PAL
$00 $02 $04 $06 $08 $0A $0C $0E
$00
$10
$20
$30
$40
$50
$60
$70
$80
$90
$A0
$B0
$C0
$D0
$E0
$F0


SECAM
$00 $02 $04 $06 $08 $0A $0C $0E
$00
$10
$20
$30
$40
$50
$60
$70
$80
$90
$A0
$B0
$C0
$D0
$E0
$F0

Playfield graphics (1)

Resolution: 40 bits – 4 color clock cycles per bit

Registers responsible for playfield generation:

COLUPF: color

PF0, PF1, PF2: data

How to squeeze this 40 bit resolution into 3 bytes?

CTRLPF: control register

- Bit 0: 1=reflect playfield, 0=repeat playfield
- Bit 1: 1=use player colors, 0=use playfield color
- Bit 2: 1=playfield over sprites, 0=sprites over playfield

Playfield graphics (2)

The data registers in depth:

- PF0: ABCD ----
- PF1: EFGH IJKL
- PF2: MNOP QRST

So the playfield data are only 20 bits that can be

Mirrored: DCBAEFGHIJKLTSRQPONMMNOPQRSTLKJIHGFEABCD

Repeated: DCBAEFGHIJKLTSRQPONMDCBAEFGHIJKLTSRQPONM

Changed: DCBAEFGHIJKLTSRQPONMdcbaefghijkltsrqponm

Note: Intuitive and straight forward to code for, well this isn't

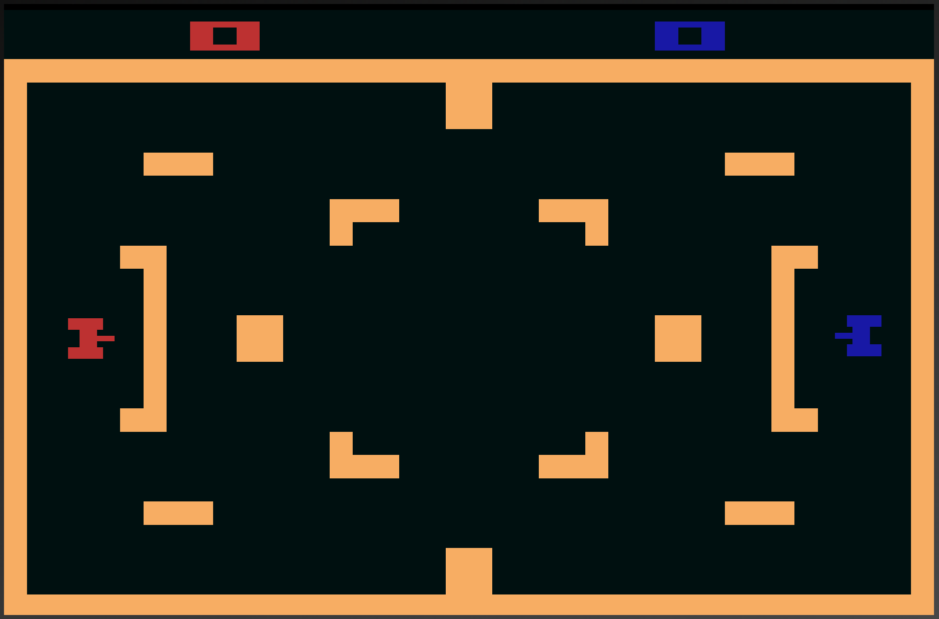
Real life playfield examples

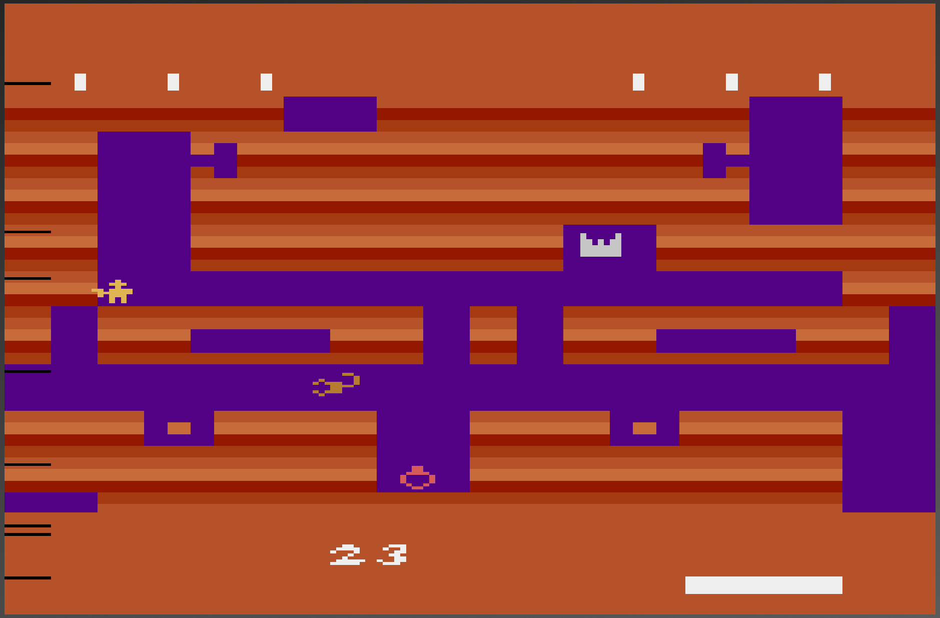
Examples from games:

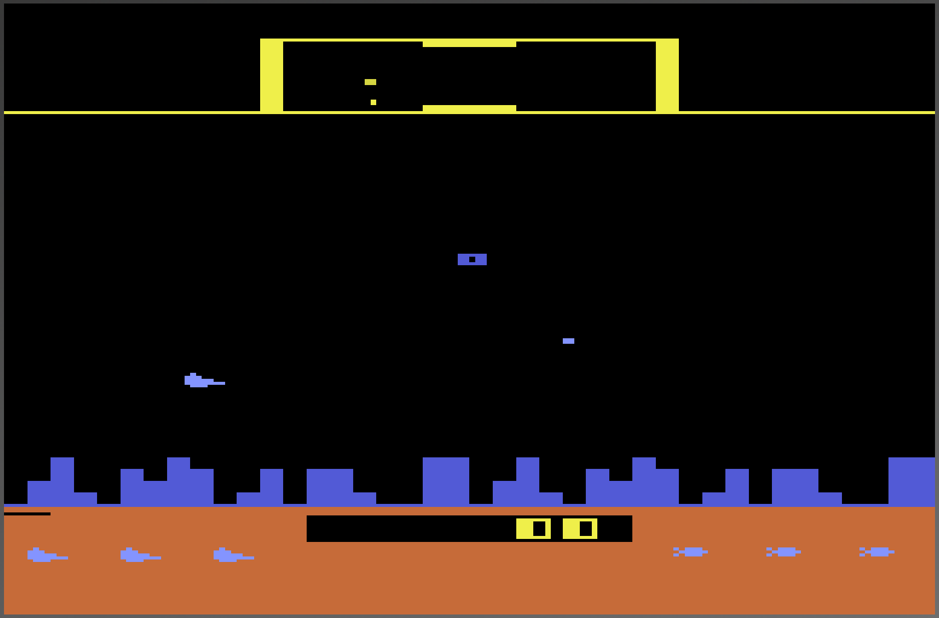
Combat (mirrored)

Defender (repeated)

Tutankham (alternating)







Sprites

The TIA has 5 sprites:

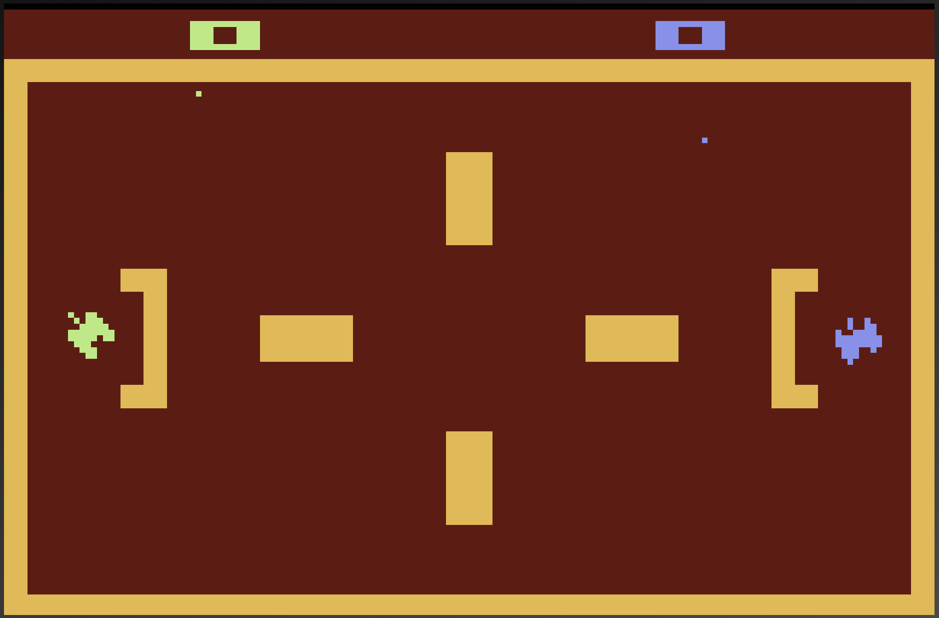
– 2 player sprites (8 bit data)
– 2 missile sprites (1 bit on/off)
– 1 ball sprite (1 bit on/off)

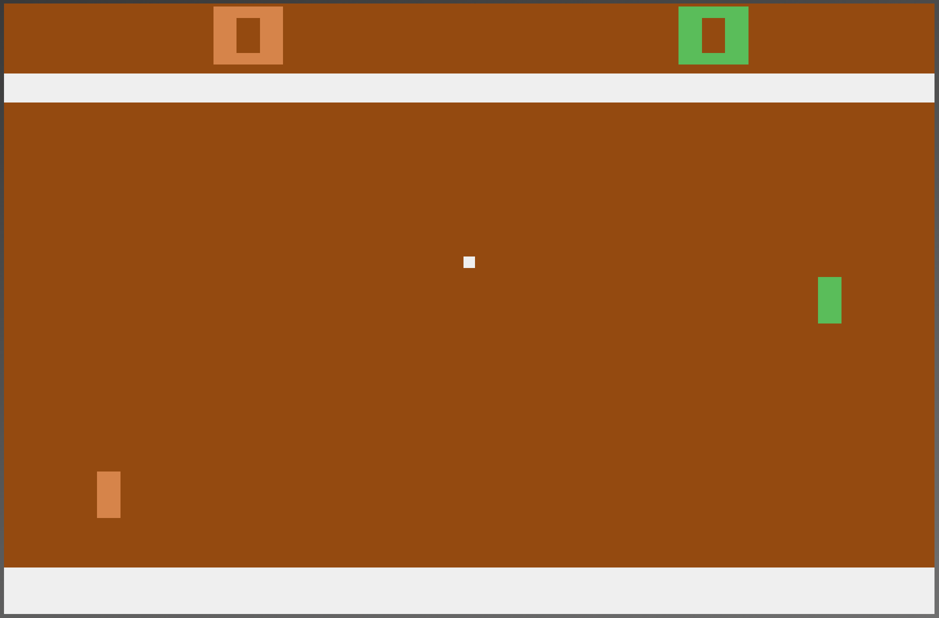
Missile sprite positions can be linked to player positions or positioned independently

Hardware was designed for running

Combat

Pong (Video Olympics)





Sprites placement (1)

How are sprites placed on the screen?

Y: enable before beam reaches position

X: more complicated, though

RESP0, RESP1, RESM0, RESM1, RESBL

Reset the sprite position, no value taken

"Reset" has a slightly different interpretation here:

Not reset to position 0, but to current X position of beam

Sprites placement (2)

TIA clock 3 times as fast as CPU clock

Fine-tuning the position:

HMP0, HMP1, HMM0, HMM1, HMBL:

- 4 bit signed motion register
- can move -8 to +7 color clock cycles
- negative moves right, positive left

HMOVE:

- apply motion register settings

HMCLR:

- clear all HMxx registers at once

Keeping in sync

Since the timing of writing to the registers is essential, it is crucial to know where the beam is

To accomplish this, there are three rules:

1) Count the cycles: of every opcode the time it takes to execute is known

2) Use a write to WSYNC to stop the CPU until the start of a new scanline is reached

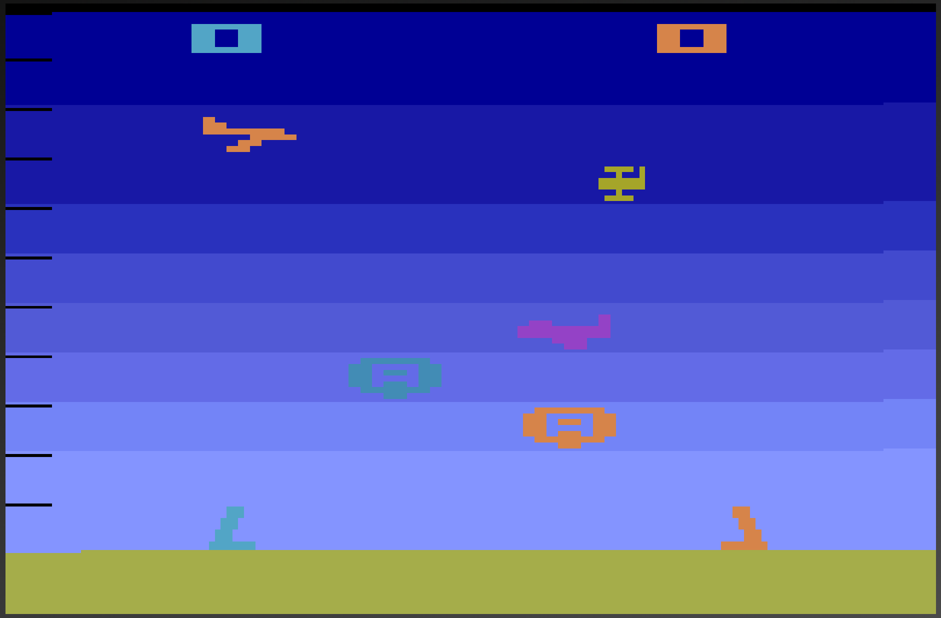
3) If you can't predict how long some code will take, start the timer and wait for it to timeout after the work is done

Real life sprites examples (1)

How get more sprites?

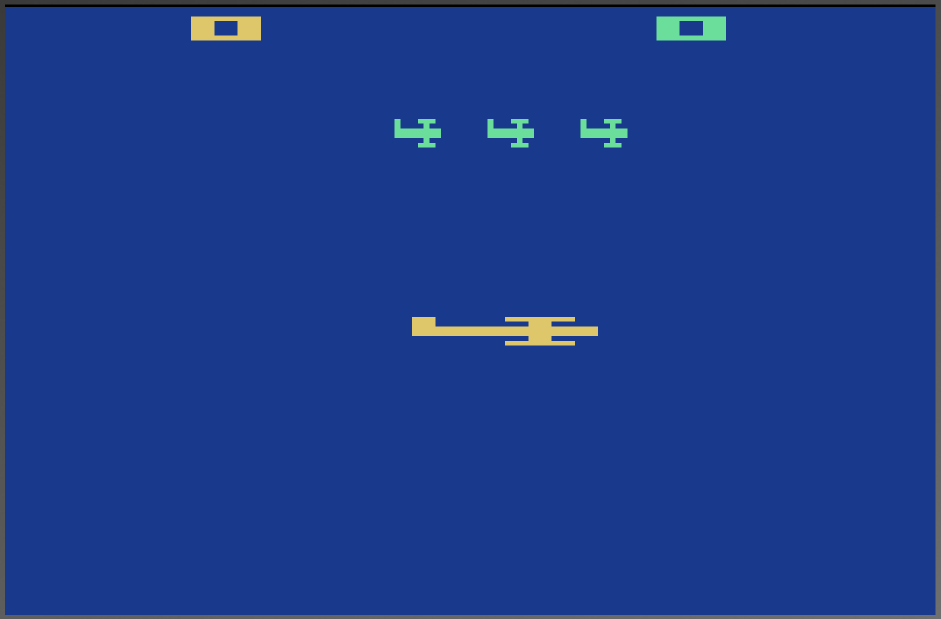
Can be done in software

Air-Sea Battle (1977)



Hardware helps a bit

Combat (1977)

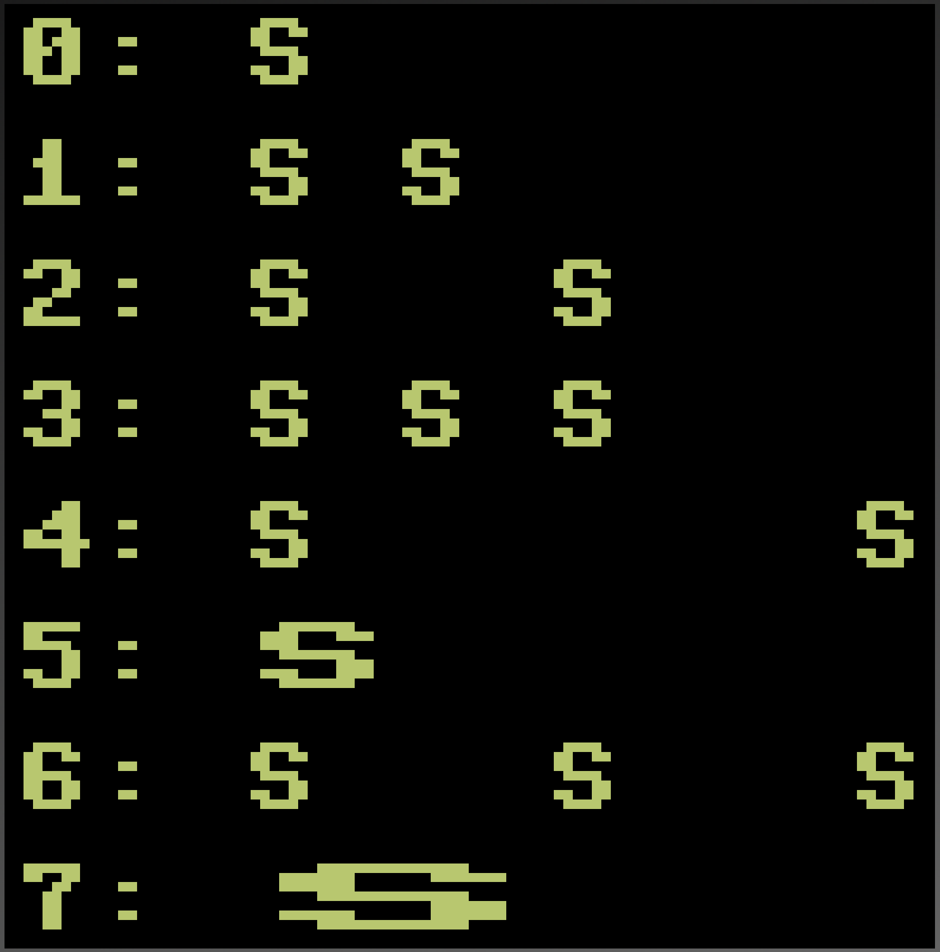


Sprites: size and repetition

The player sprites can be repeated or stretched in 7 different ways

Mirroring of player sprites is also possible

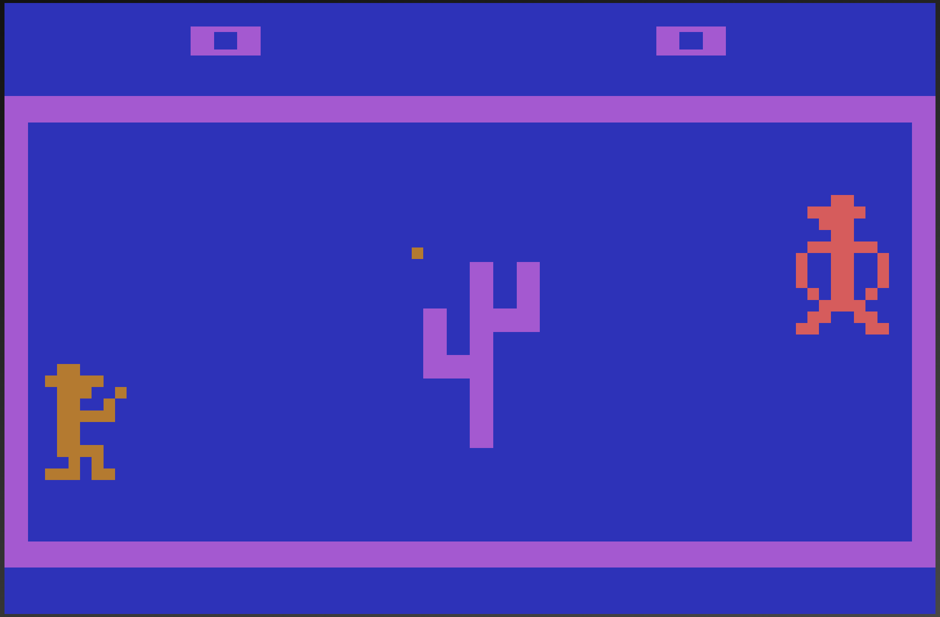
Ball and missile sprites can be defined being in size of 1, 2, 4 or 8 clock cycles



```
0: S
1: S S
2: S   S
3: S S S
4: S       S
5: S
6: S   S   S
7: S
```

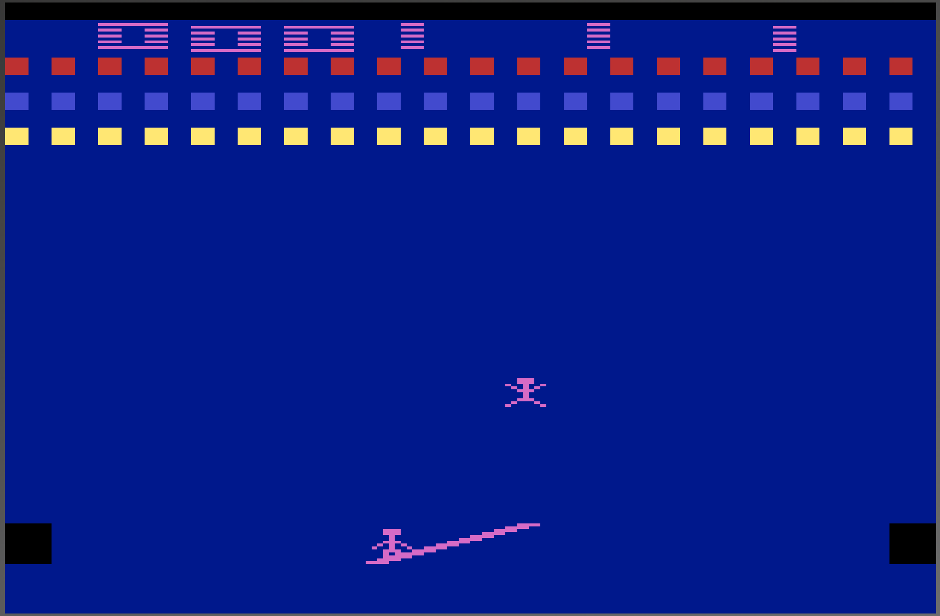
Real life sprite examples (2)

Outlaw (1978):



2 player sprites
2 missile sprites

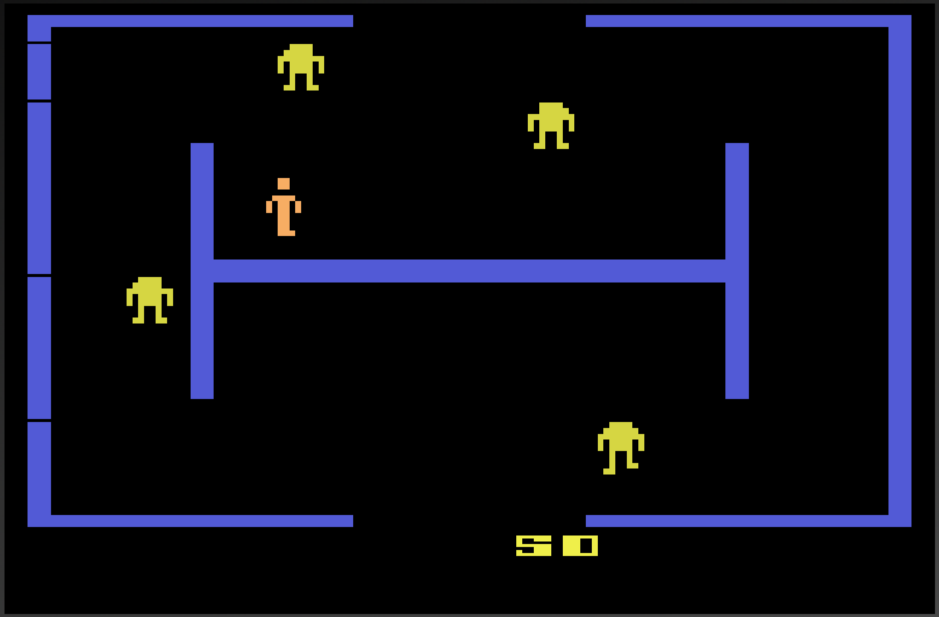
Circus Atari (1978):



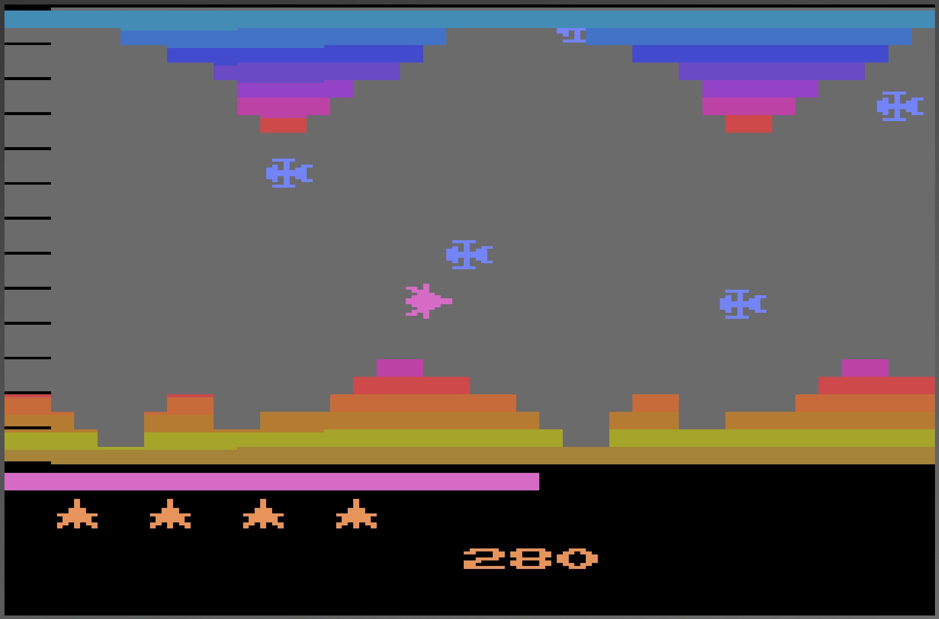
Both player sprites used for clowns, seesaw is a missile moved half of its size each scanline

Real life sprites examples (3)

Berzerk (1982):



Vanguard (1982):

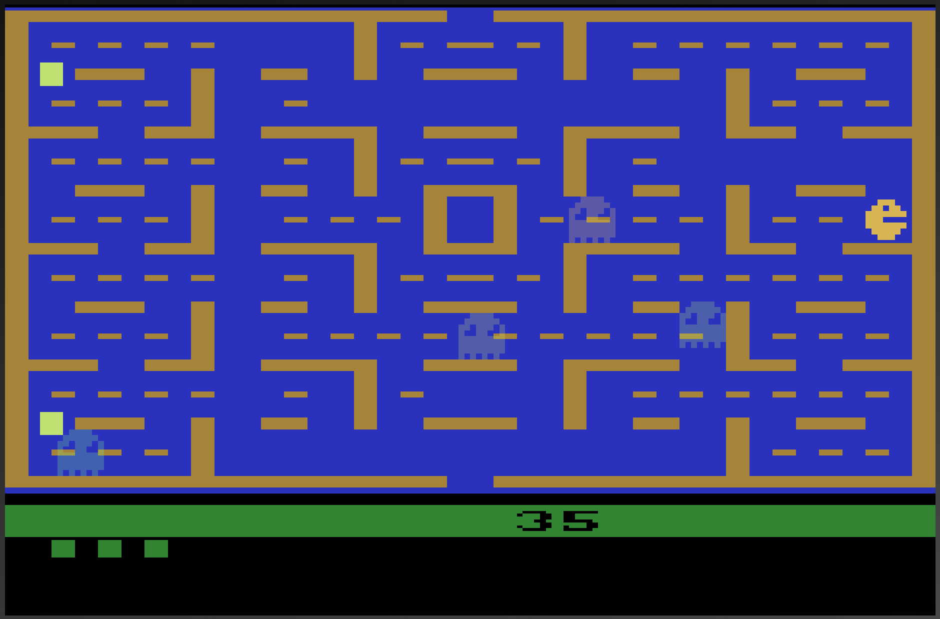


Both make sure in gameplay that enemies are not on the same scanline.

Real life sprites examples (4)

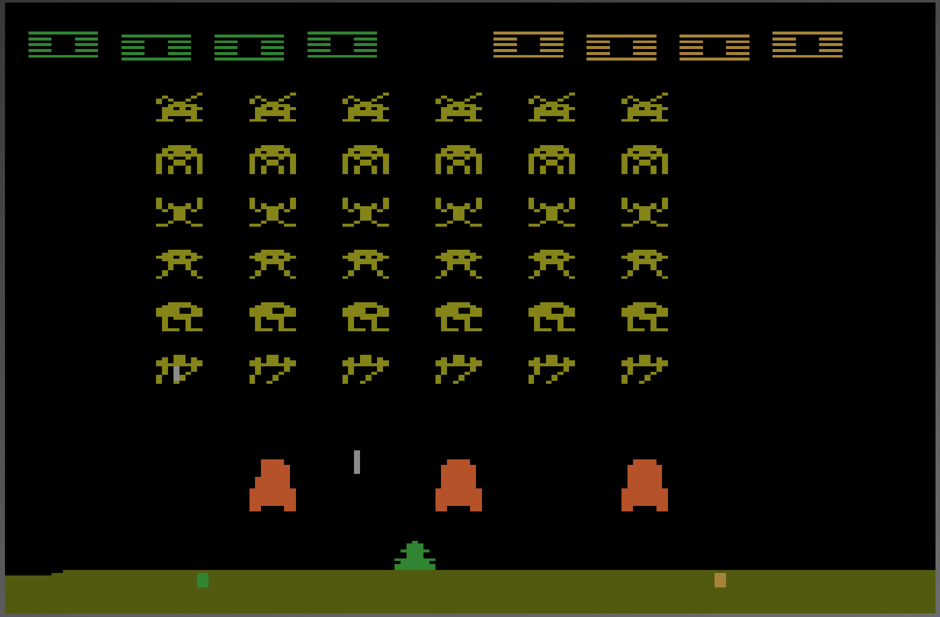
Pac-Man (1981):

Uses interlace: only one of the 4 ghosts is drawn per frame

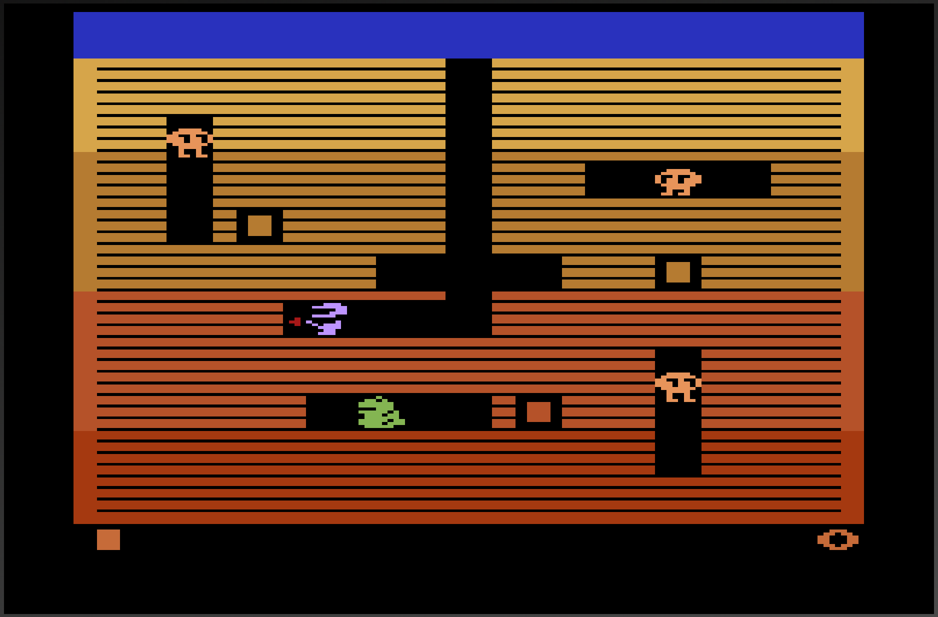


Space Invaders (1978):

Uses sprite triplication for both sprites to draw aliens

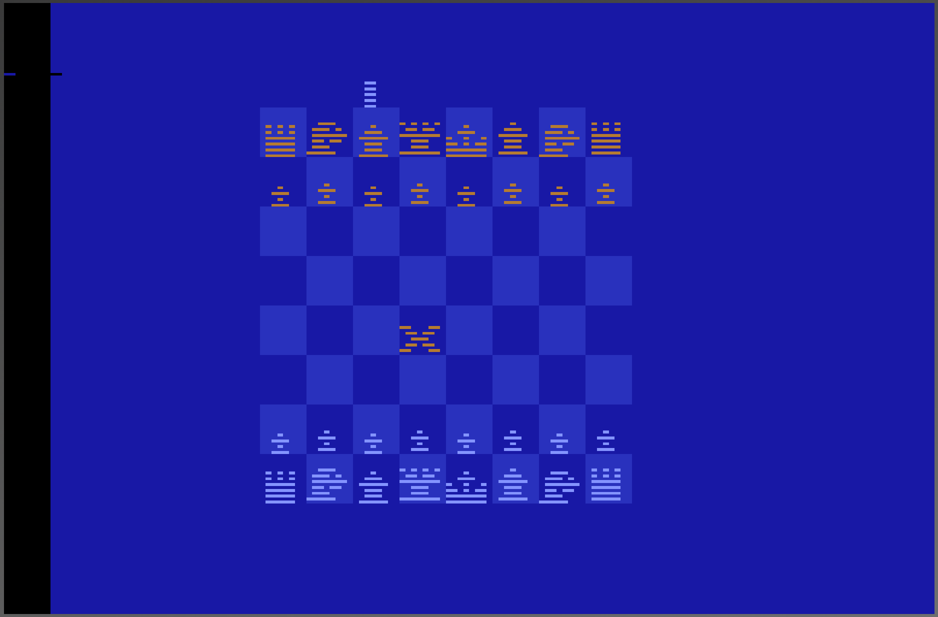


Real life sprites examples (5)



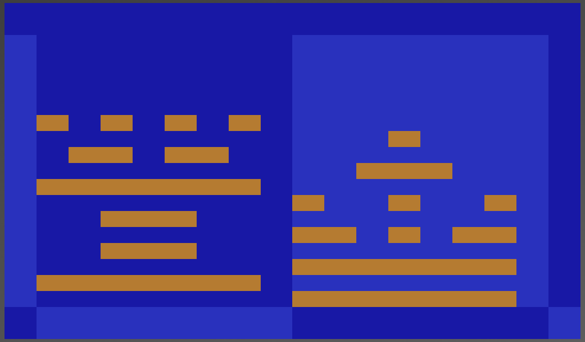
Dig Dug (1983):

Uses interlace only when more than 2 sprites are on the same scan line



Video Chess (1978):

Draws sprites only every other line



Detecting collisions (1)

Collision detection is essential for gameplay

Hardware is full featured here

The read registers of the TIA:

CXM0P CXM1P CXP0FB CXP1FB CXM0FB CXM1FB

CXBLPF CXPPMM INPT0 INPT1 INPT2 INPT3

INPT4 INPT5

Collision Controller

8 registers for collision detection (15 bits used)
6 registers for controller input

Registers will keep bits until CXCLR is triggered

Detecting collisions (2)

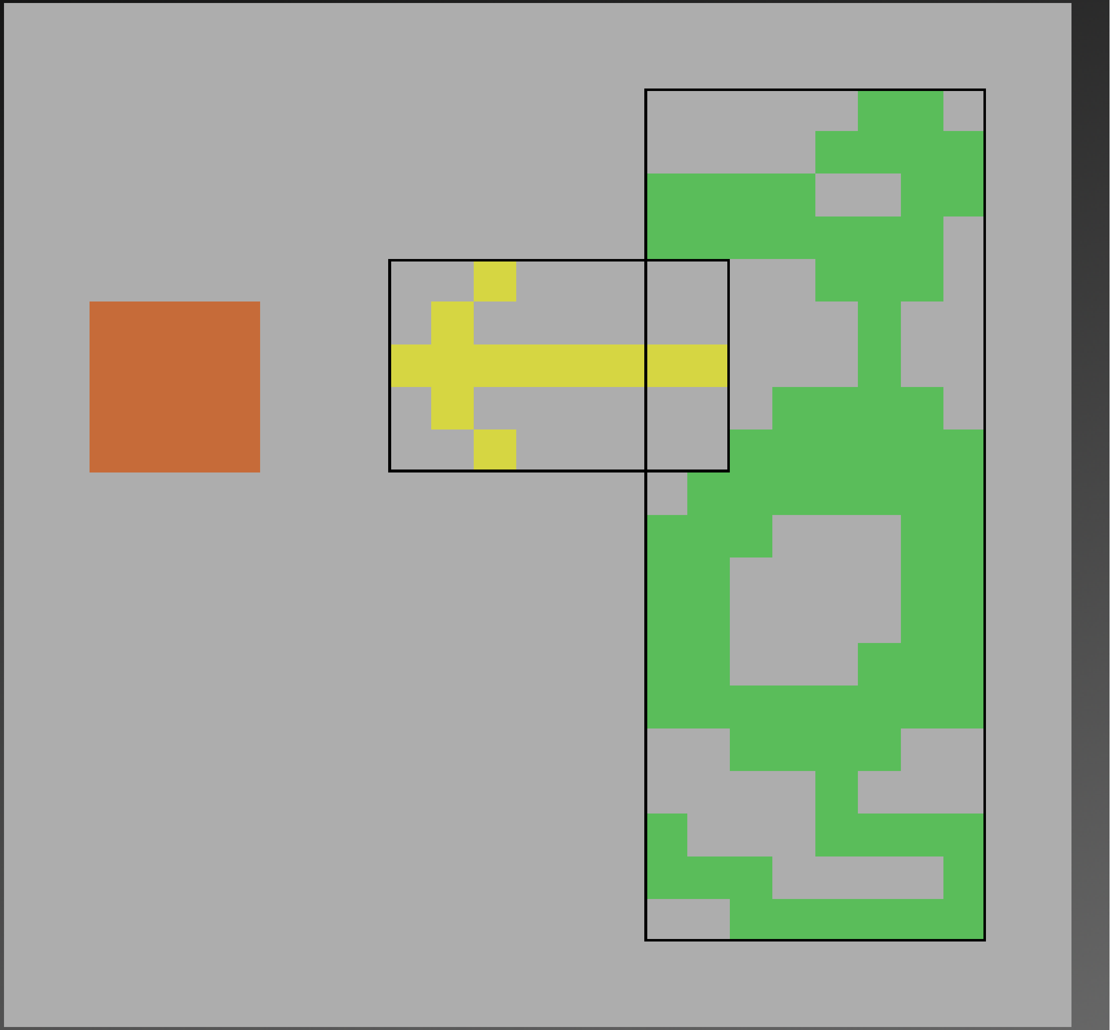
What is a collision?

There are two options:

A: when pixels touch

B: when drawing areas touch

On the TIA the correct answer is A, so this is not a collision



Adventure (1978)

Audio (1)

The TIA has 2 voices each having 3 registers

AUDV0, AUDV1: Volume 4 bit

AUDF0, AUDF1: Frequency 5 bit

- Base frequency divided by (AUDFx + 1)

AUDC0, AUDC1: Control 4 bit

- 11 unique settings
- Most of the settings are not used for music,
  but for sound effects like motor noise, shots, ufos...

Audio (2)

Sound generation can be looked at in two steps:

Step 1: basic signal is generated by setting the audio line high or low: basic output is a rectangle

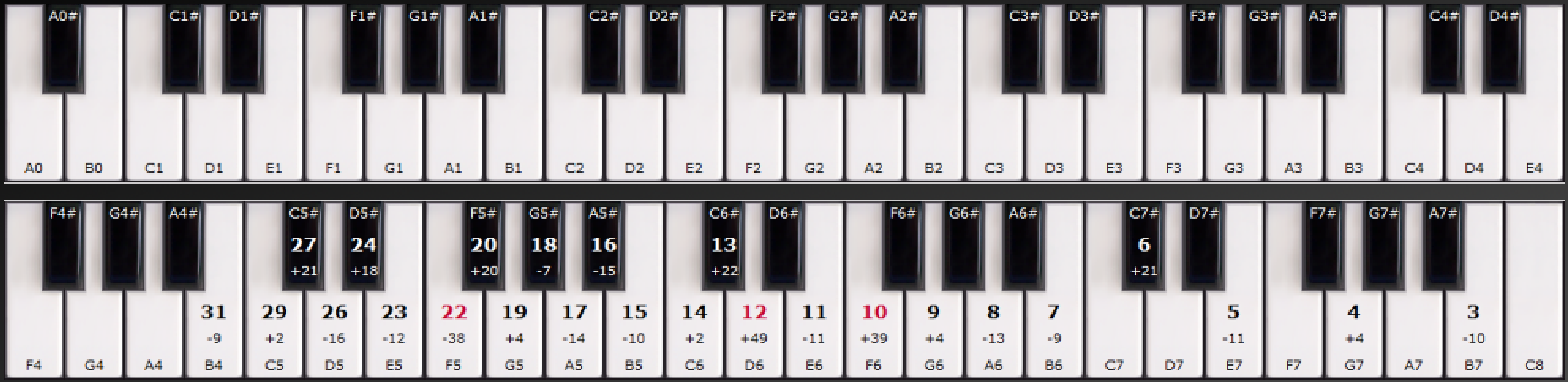
Base frequency = color clock / 114
NTSC: 3579575 Hz / 114 = 31399.78 Hz
PAL: 3546894 Hz / 114 = 31113.10 Hz

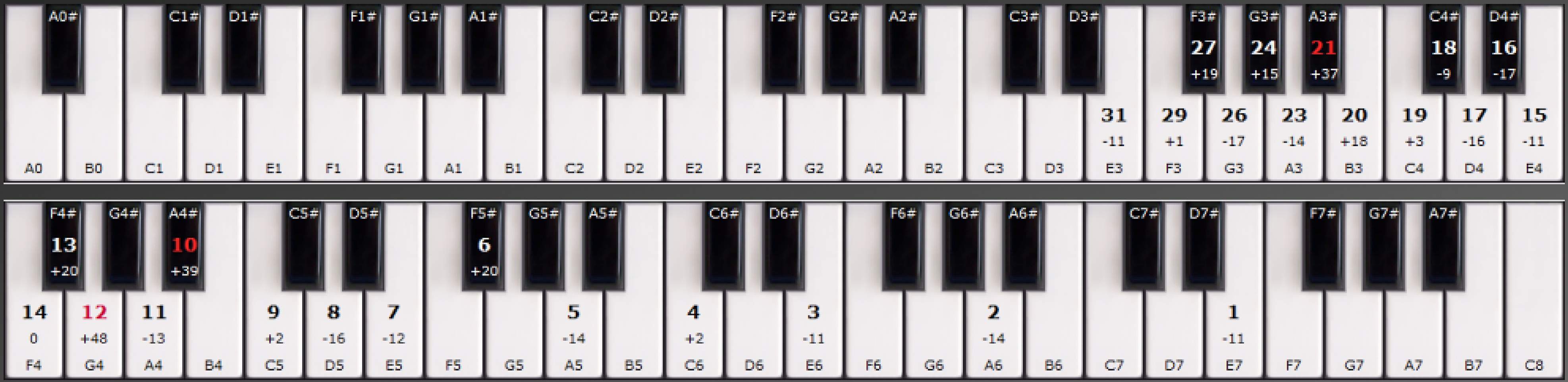
AUDC0, AUDC1 define the bit pattern

Sound generated by shifting out the bit pattern

Audio (3)

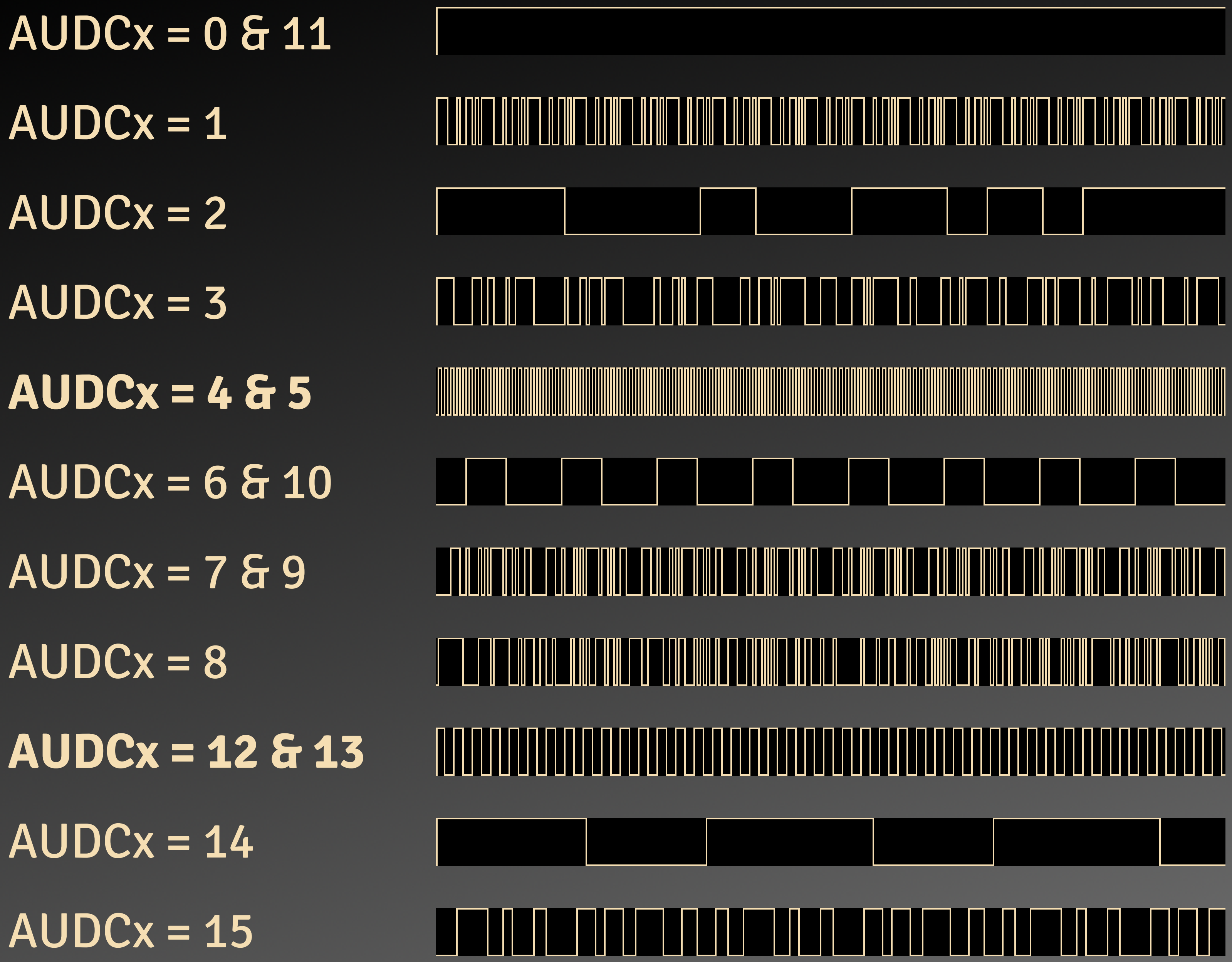
AUDCx: keys for settings 4 and 12


A0# C1# D1# F1# G1# A1# C2# D2# F2# G2# A2# C3# D3# F3# G3# A3# C4# D4#
A0 B0 C1 D1 E1 F1 G1 A1 B1 C2 D2 E2 F2 G2 A2 B2 C3 D3 E3 F3 G3 A3 B3 C4 D4 E4
F4# G4# A4# C5# D5# F5# G5# A5# C6# D6# F6# G6# A6# C7# D7# F7# G7# A7#
C5# 27 +21
D5# 24 +18
F5# 20 +20
G5# 18 -7
A5# 16 -15
C6# 13 +22
C7# 6 +21
B4 31 -9
C5 29 +2
D5 26 -16
E5 23 -12
F5 22 -38
G5 19 +4
A5 17 -14
B5 15 -10
C6 14 +2
D6 12 +49
E6 11 -11
F6 10 +39
G6 9 +4
A6 8 -13
B6 7 -9
E7 5 -11
G7 4 +4
B7 3 -10
F4 G4 A4 B4 C5 D5 E5 F5 G5 A5 B5 C6 D6 E6 F6 G6 A6 B6 C7 D7 E7 F7 G7 A7 B7 C8


A0# C1# D1# F1# G1# A1# C2# D2# F2# G2# A2# C3# D3# F3# G3# A3# C4# D4#
F3# 27 +19
G3# 24 +15
A3# 21 +37
C4# 18 -9
D4# 16 -17
E3 31 -11
F3 29 +1
G3 26 -17
A3 23 -14
B3 20 +18
C4 19 +3
D4 17 -16
E4 15 -11
A0 B0 C1 D1 E1 F1 G1 A1 B1 C2 D2 E2 F2 G2 A2 B2 C3 D3 E3 F3 G3 A3 B3 C4 D4 E4
F4# G4# A4# C5# D5# F5# G5# A5# C6# D6# F6# G6# A6# C7# D7# F7# G7# A7#
F4# 13 +20
A4# 10 +39
F5# 6 +20
F4 14 0
G4 12 +48
A4 11 -13
C5 9 +2
D5 8 -16
E5 7 -12
A5 5 -14
C6 4 +2
E6 3 -11
A6 2 -14
E7 1 -11
F4 G4 A4 B4 C5 D5 E5 F5 G5 A5 B5 C6 D6 E6 F6 G6 A6 B6 C7 D7 E7 F7 G7 A7 B7 C8

Images courtesy of www.randomterrain.com, used by permission

Possible basic waveforms



AUDCx = 0 & 11

AUDCx = 1

AUDCx = 2

AUDCx = 3

AUDCx = 4 & 5

AUDCx = 6 & 10

AUDCx = 7 & 9

AUDCx = 8

AUDCx = 12 & 13

AUDCx = 14

AUDCx = 15

Audio (4)

Step 2: basic signal is multiplied with AUDVx

AUDCx bit pattern "0" is useful: when activated 4 bit digital audio can be played by writing data to the corresponding volume register AUDVx.

Most impressive example for this kind of sound generation is Berzerk VE (Voice Enhanced), a hack which features the voice of the arcade version!

Next steps (1)

Play a game!

Get an emulator, games are available for download

I have not covered any homebrew games, leaving them for you to discover

There are a lot of them, and they are usually "better coded" than most of the games from the '70s or '80s

Take a look at them and try to figure them out with what you've learned here

Next steps (2)

Play with the system: code something!

Tools are available for free

A lot of examples for different tricks are around

Stella has excellent debugging support

If you already know 6502 assembler, something like a playfield scroller can be coded in an afternoon

Next steps (3)

Prepare a talk for the next congress!

There are a lot of other cool systems that we would like to learn about: (these are just suggestions)

Consoles:

- ColocoVision
- Game Boy
- Game Boy Advance
- Intellivision
- NES (Famicom)
- SNES (Super Famicom)

Computers:

- Amiga
- Amstrad CPC (Schneider)
- Apple II
- Atari 400/800 XL
- Spectrum
- ZX-81

Thank you for your attention!

Questions?