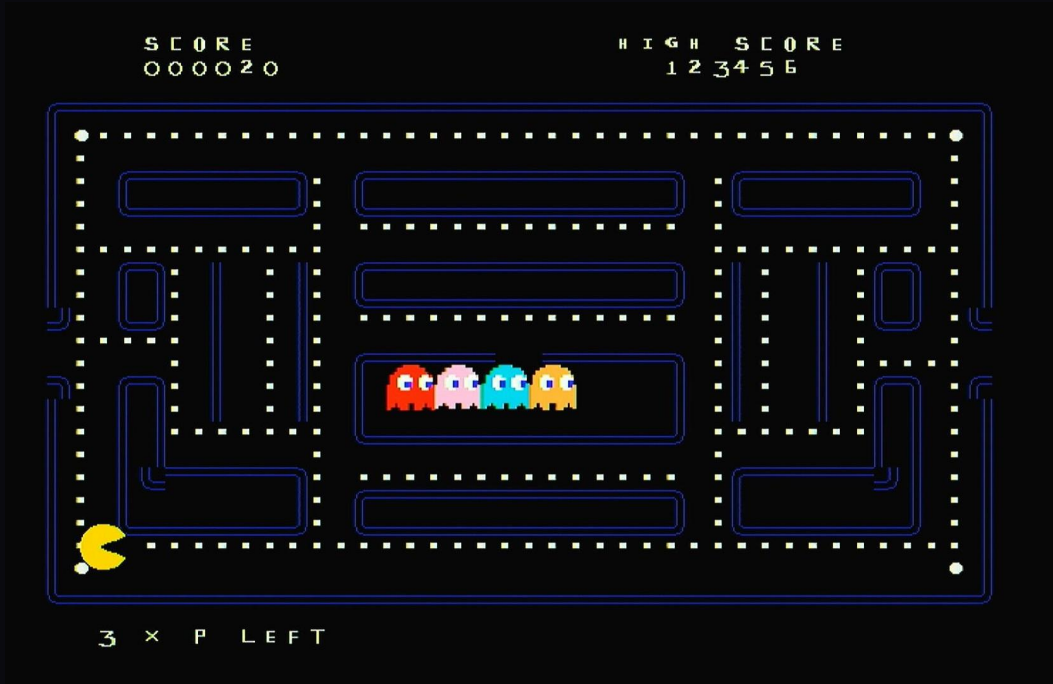
Pac-Man on the Cyclone V SoC

CSEE 4840 Embedded Systems Design | Austin Gnecco, Levi Sharma, Alessa Merkudanova and Connor Marvin



System Architecture

HPS - SOFTWARE (ARM)

- Linux OS
- USB Controller Poll
- Game State Loop (AI/Physics)
- Userspace Sound Mixer
- VGA Kernel Driver (vga_pacman.ko)
- Audio Kernel Driver (pacman_audio.ko)

AVALON BRIDGE

FPGA - HARDWARE

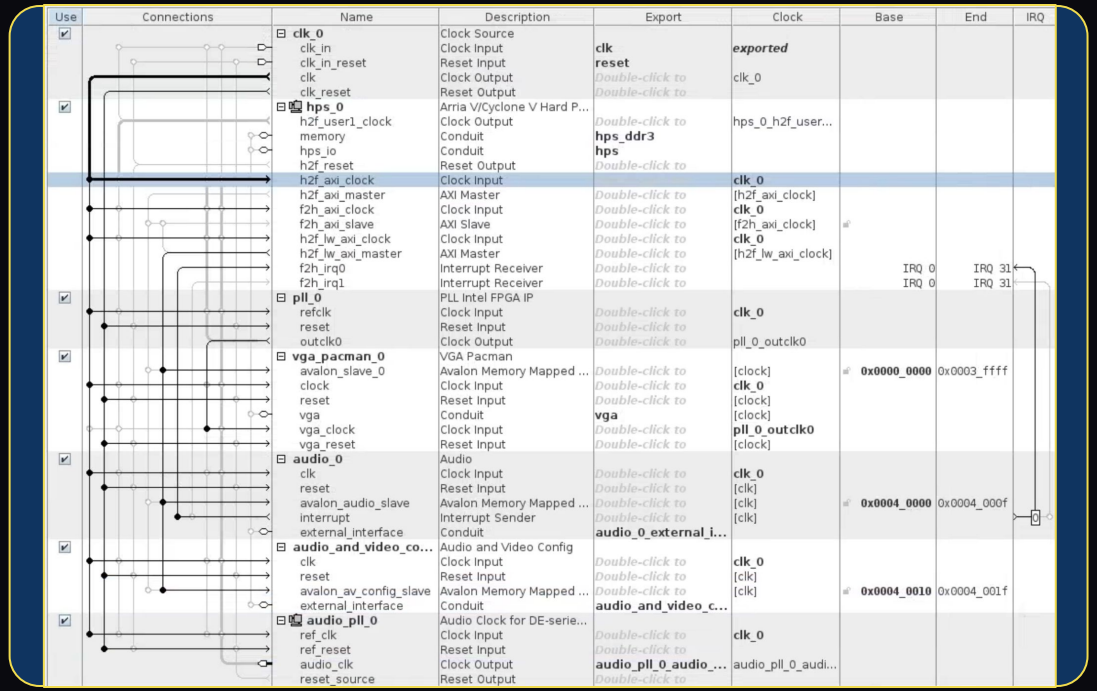
- Tileset/Sprite Character ROMs
- VGA Controller
- Wolfson Audio Chip Interface

Hardware (FPGA Fabric): Contains the tileset memory blocks, sprite palette lookup, VGA controller and external Audio Codec control.

Software (Hard Processor System): Runs Embedded Linux on a dual-core ARM Cortex-A9, coordinating game physics, AI pathfinding, decoding the controller inputs, and sound mixing.

Interconnect: Intel Avalon Lightweight and High-Speed HPS-to-FPGA Memory Mapped Bridges.

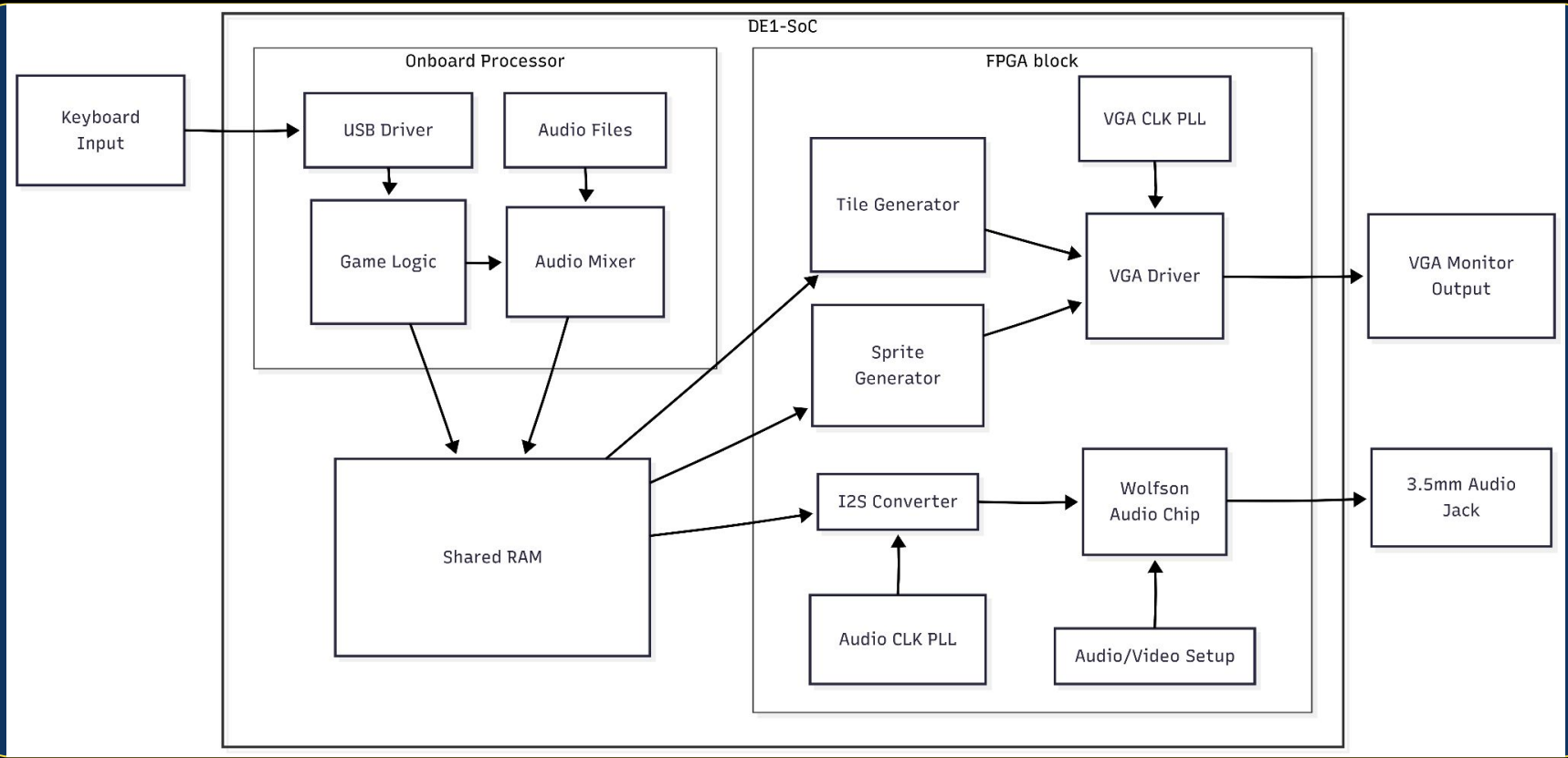
Hardware Setup in Quartus



CORE QSYS MODULES

- vga_pll0: Provides the ~25MHz clock required to drive VGA output
- Vga_driver: Controls writing sprite positions and frames, as well as tilemap changes.
- audio_pll0: Provides the audio_0 clock
- Audio0: Contains FIFOs that are filled off of the Avalon Memory Bus and create an I2C stream for the Wolfson chip.
- Audio_and_video_config: Creates a configuration commands over I2C to the Wolfson Chip

Block Diagram



Register Map

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VGA Graphics Block (Base 0xff200000)																
0x0000							Px9	Px8	Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0
0x0001								Py8	Py7	Py6	Py5	Py4	Py3	Py2	Py1	Py0
0x0002											Pi5	Pi4	Pi3	Pi2	Pi1	Pi0
0x0003																ready
...	*Unused / Reserved Address Space*															
0x0010							G0x9	G0x8	G0x7	G0x6	G0x5	G0x4	G0x3	G0x2	G0x1	G0x0
0x0011								G0y8	G0y7	G0y6	G0y5	G0y4	G0y3	G0y2	G0y1	G0y0
0x0012									G0s1	G0s0	G0i5	G0i4	G0i3	G0i2	G0i1	G0i0
...	*Unused Sprite Registers*															
0x0014							G1x9	G1x8	G1x7	G1x6	G1x5	G1x4	G1x3	G1x2	G1x1	G1x0
0x0015								G1y8	G1y7	G1y6	G1y5	G1y4	G1y3	G1y2	G1y1	G1y0
0x0016									G1s1	G1s0	G1i5	G1i4	G1i3	G1i2	G1i1	G1i0
...	*Unused Sprite Registers*															
0x0018							G2x9	G2x8	G2x7	G2x6	G2x5	G2x4	G2x3	G2x2	G2x1	G2x0
0x0019								G2y8	G2y7	G2y6	G2y5	G2y4	G2y3	G2y2	G2y1	G2y0
0x001A									G2s1	G2s0	G2i5	G2i4	G2i3	G2i2	G2i1	G2i0
...	*Unused Sprite Registers*															
0x001C							G3x9	G3x8	G3x7	G3x6	G3x5	G3x4	G3x3	G3x2	G3x1	G3x0
0x001D								G3y8	G3y7	G3y6	G3y5	G3y4	G3y3	G3y2	G3y1	G3y0
0x001E									G3s1	G3s0	G3i5	G3i4	G3i3	G3i2	G3i1	G3i0
...	*Unused / Reserved Address Space*															
0x0100									r0c0i7	r0c0i6	r0c0i5	r0c0i4	r0c0i3	r0c0i2	r0c0i1	r0c0i0
0x0101									r0c1i7	r0c1i6	r0c1i5	r0c1i4	r0c1i3	r0c1i2	r0c1i1	r0c1i0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x0826									r29c38i7	r29c38i6	r29c38i5	r29c38i4	r29c38i3	r29c38i2	r29c38i1	r29c38i0
0x0827									r29c39i7	r29c39i6	r29c39i5	r29c39i4	r29c39i3	r29c39i2	r29c39i1	r29c39i0
Altera UP Audio Core Block (Base 0xff240000)																
Audio 0x00													CW	CR	WE	RE
Audio 0x01	RR7	RR6	RR5	RR4	RR3	RR2	RR1	RR0	RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0
Audio 0x02	WL7	WL6	WL5	WL4	WL3	WL2	WL1	WL0	WR7	WR6	WR5	WR4	WR3	WR2	WR1	WR0
Audio 0x03	L15	L14	L13	L12	L11	L10	L9	L8	L7	L6	L5	L4	L3	L2	L1	L0
Audio 0x04	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0

Legend

```
Px / Py   : Pac-Man X (10-bit) / Y (9-bit) pixel pos.
Pi        : Pac-Man (0=visible, 2:1=rotate, 5:3=img.).
ready     : VGA timing and sync bit.
G[0-3]x/y: Ghost 0-3 X(10-bit) / Y(9-bit) pixel pos.
G[0-3]i   : Ghost 0-3(0=visible, bits 2:1=rotate,
5:3=img.).
G[0-3]s   : Ghost 0-3 State (00=Normal color,
01=Vulnerable blue, 10=Eaten eyes).
rXcY      : Tile coordinates (Row 0-29, Column 0-39).
rXcYi     : 8-bit Tile ID at coordinates (wall, empty,
food pellet, cherry, etc.).
RE / WE   : Audio Read/Write FIFO hardware interrupt
enable flags.
CR / CW   : Audio Read / Write FIFO master clear
commands.
RL / RR   : Left / Right channel read FIFO filled
level (0-128 words).
WL / WR   : L/R channel write FIFO empty space (0-128
words).
L / R     :16-bit L/R signed PCM audio sample value.
```

VGA Graphics Engine

Name	Type	Image	Size (bits)	# of Frames	Total size
Pac-Man	Sprite		32×32	4	16,384
Ghosts	Sprite		32×32	5	20,480
Map Pieces	Tile		16×16	6	12,288
Dots	Tile		16×16	2	4,096
Characters	Tile		16×16	36	65,536

Tilemap Architecture

The game uses a 40x30 tile grid mapped to the Avalon Memory Mapped Bus, with tile designs stored on the FPGA

Tileset ROM stores the actual pixel patterns for each tile.

Sprite Generation

Sprites for Pac-Man and the 4 ghosts, are rendered on top of the tiles from dedicated sprite ROMs.

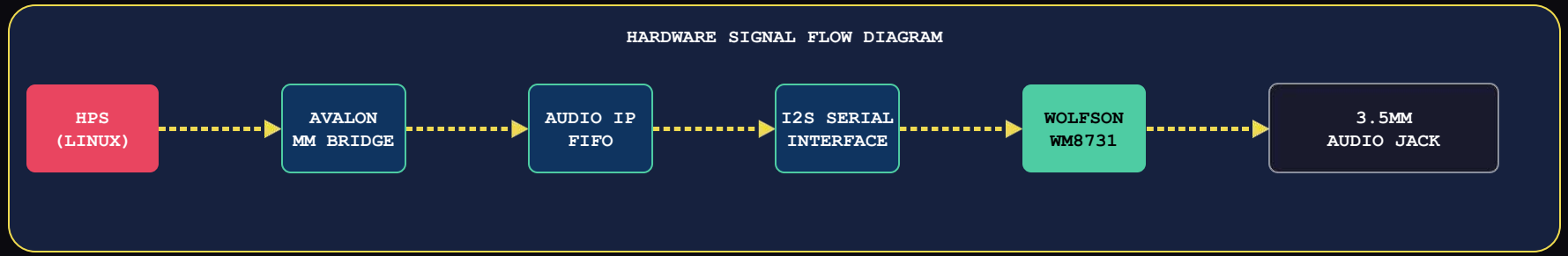
Palette ROM

Hardware translation of 4-bit indices to raw 24-bit RGB values, allowing full RGB colors with less data stored.

Smooth Motion Control

HW scroll registers allow sub-pixel rendering. Software writes pixel coordinates over `/dev/vga_pacman`.

Physical Audio Pipeline & DAC Configuration



External Codec

The Wolfson WM8731 audio CODEC chip takes the data passed from the HPS and converts it to analog audio output

Autonomous Initialization

Custom FPGA block writes over physical I2C lines to initialize sample rate to 48KHz at power-up.

Hardware FIFO Buffer

Dual-channel circular buffer in FPGA fabric to smooth sample delivery jitter.

Audio Clock PLL

Synthesized hardware module outputs a stable 12.5MHz master clock to prevent drift.

Game Software Logic

Pac-Man and Ghost Game Logic

- Calculations rely on 40x30 matrices that are used for path verification
- Ghosts use matrices to figure out where Pacman is and how to move based on their algorithms
- Pacman uses his matrix to determine if he can move in the direction he currently is going in
- Matrices are ideal for software since they are inexpensive computationally and allow for much easier logic implementation

Pac-Man and Ghost Movement Schemes

- Pac-Man: relies on user input and available paths in the maze grid.
- Blinky: directly targets Pac-Man's current grid position.
- Pinky: targets a point several tiles ahead of Pac-Man's current direction.
- Inky: uses both Pac-Man's forward position and Blinky's position to calculate a projected target.
- Clyde: chases Pac-Man when far away, but retreats toward his corner when he gets too close.

Keyboard and Controller Input

- Two types of usable user input which are controller and keyboard
- User input is polled faster than the visual display update, which makes movement changes feel more responsive.
- The game uses libusb to read the user input directly instead of relying on a higher-level input system.
- USB packets are inspected and decoded manually, so the software knows exactly which byte patterns correspond to up, down, left, and right
- With the usage of packet decoding the program understands which bytes within the packets indicate a change in direction which allows the program to run even with unknown button presses

Terminal Screenshots

Ghosts Chasing Pacman

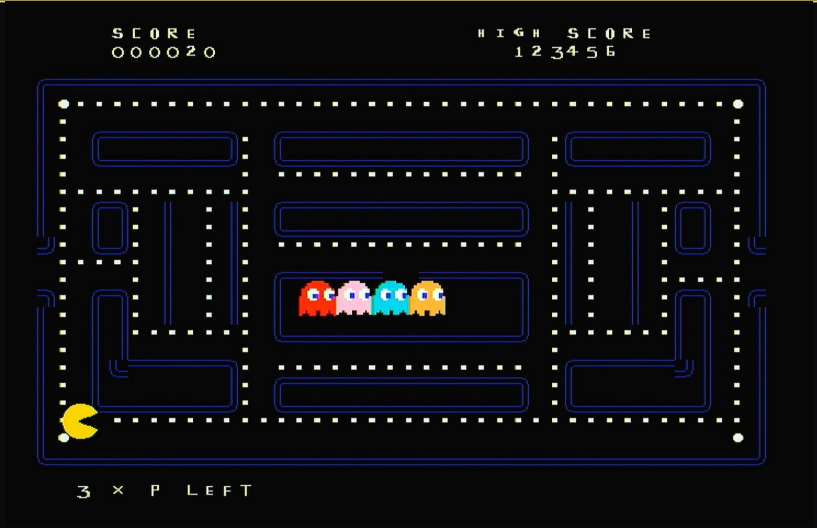
```
Pac-Man game running on /dev/vga_pacman
Input: keyboard WASD/arrows | Score=420 | High=123456 | Lives=3 | Pellets=38/250 | Normal=37 | Super=1 | Ghosts=0
Tick=1025 | Power ticks left=0 | Render=20000us | Pac=72000us | Ghost=250000us | USB polls=3 | Win=0
Pac=(37,23) Dir=RIGHT Next=RIGHT Anim=72000/72000 LastInput=none/NONE
Blinky=(27,12) T=(37,23) DOWN CHASE leave=0 | Pinky=(27,12) T=(41,23) DOWN CHASE leave=0
Inky=(27,12) T=(51,35) DOWN CHASE leave=0 | Clyde=(27,12) T=(37,23) DOWN CHASE leave=0
USB packet:
Controller responsiveness improved: 1ms USB timeout, 3 polls/render loop. q quits.
Controller responsiveness improved: 1ms USB timeout, 3 polls/render loop. q quits.
eave=0
USB packet:
Controller responsiveness improved: 1ms USB timeout, 3 polls/render loop. q quit
s.
```

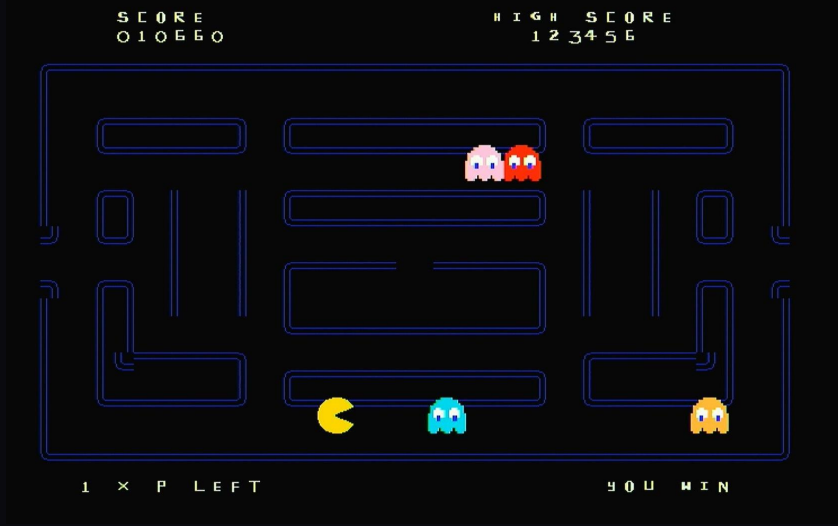
Ghosts Fleeing From Pacman

```
Pac-Man game running on /dev/vga_pacman
Input: keyboard WASD/arrows | Score=480 | High=123456 | Lives=0 | Pellets=40/250 | Normal=38 | Super=2 | Ghosts=0
Tick=4080 | Power ticks left=352 | Render=20000us | Pac=72000us | Ghost=250000us | USB polls=3 | Win=0
Pac=(1,23) Dir=LEFT Next=LEFT Anim=72000/72000 LastInput=arrow-left/LEFT
Blinky=(27,12) T=(37,5) UP FLEE leave=0 | Pinky=(27,10) T=(37,5) UP FLEE leave=0
Inky=(27,10) T=(37,5) UP FLEE leave=0 | Clyde=(27,10) T=(37,5) UP FLEE leave=0
USB packet:
Controller responsiveness improved: 1ms USB timeout, 3 polls/render loop. q quits.
Controller responsiveness improved: 1ms USB timeout, 3 polls/render loop. q quits.
eave=0
USB packet:
Controller responsiveness improved: 1ms USB timeout, 3 polls/render loop. q quit
s.
```

Game Screenshots

Starting Screen

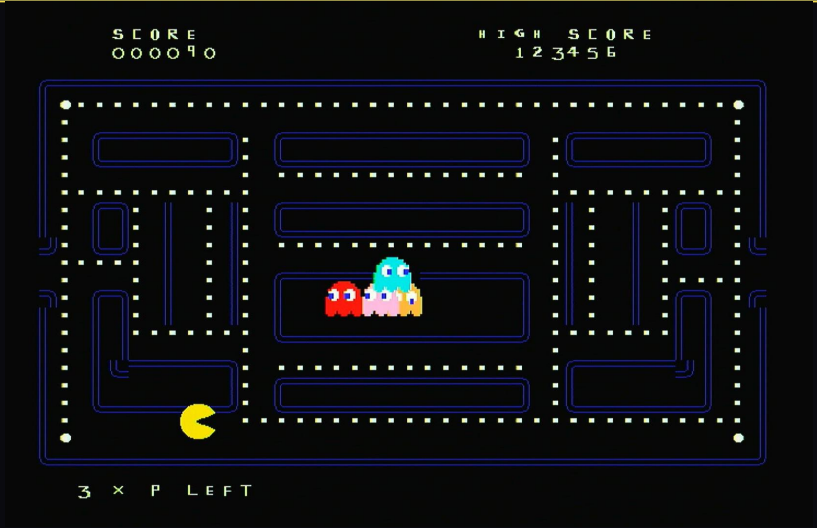


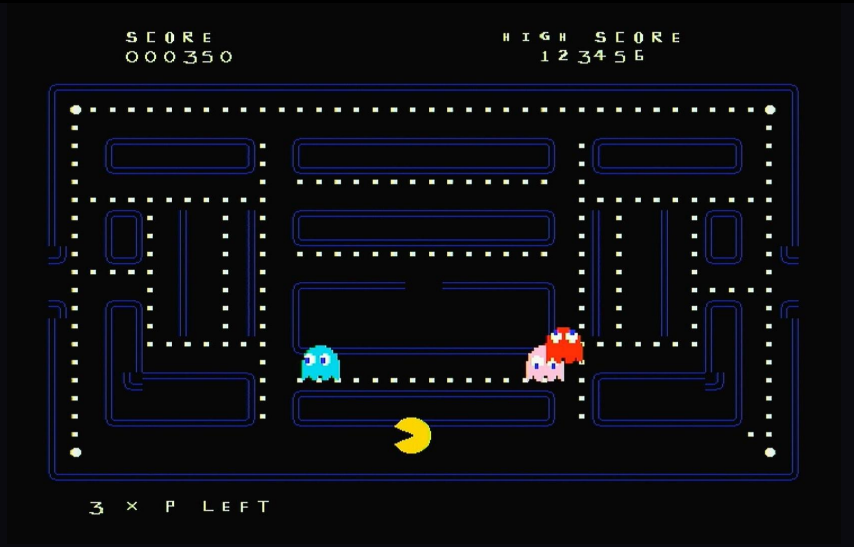


End Screen

Game Screenshots

Ghosts Leaving Spawn





Ghosts Fleeing from Pacman

Game Software Logic: Sprites and Tiles

Writing non-moving graphics

- Walls and pellets are drawn as static tile-map sprites, but only pellets change during gameplay
- Walls stay fixed once drawn and are never replaced unless the full map is redrawn
- When Pac-Man eats a pellet, that pellet tile is replaced with a blank tile
- When a pellet is cleared, the software adds that pellet's point value to the stored score integer
- The score integer is split into digits, and each digit is drawn using the matching number sprite tile

Writing moving sprites

- Pac-Man and the ghosts are not written as normal maze tiles. Pac-Man and the ghosts are written to sprite specific registers so they can move over the maze
- Pac-Man and the ghosts have direction, position, and animation-frame values, so the sprite can change while the grid logic only updates at fixed movement intervals
- Collision is checked using the 40 × 30 matrix and since Pac-Man and each ghost are 32 × 32 pixels, the code treats them as occupying a 2 × 2 tile "fake hitbox" for wall collisions, pellet eating, and ghost contact
- The code stores the previous tile and next tile, then calculates intermediate pixel positions so Pac-Man and the ghosts move smoothly across the screen