AMY STUDIO

COLECOVISION
TOOLCHAIN COMPARISON

Version November 8, 2026

Measured ROM size, runtime validation, graphics compression, capabilities, and development tradeoffs across nine ColecoVision approaches.

Six reproducible game-development fixtures. Complete source, ROM, hash, and emulator evidence.

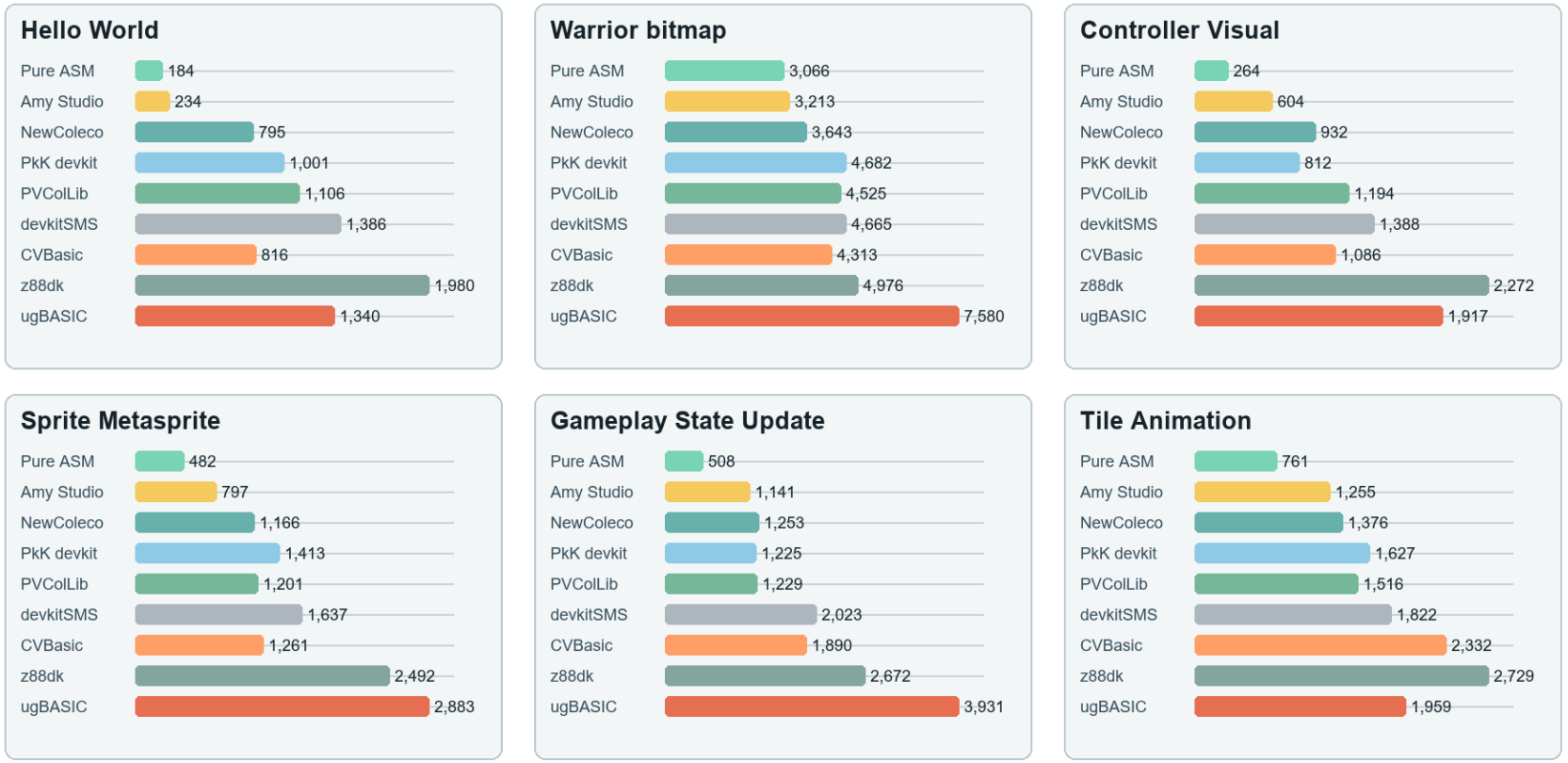
Amy Bienvenu

Six benchmark cases

Each panel uses its own scale so the relative cost of every toolchain remains readable.

Occupied ROM size by benchmark

Bytes excluding cartridge padding - lower is smaller



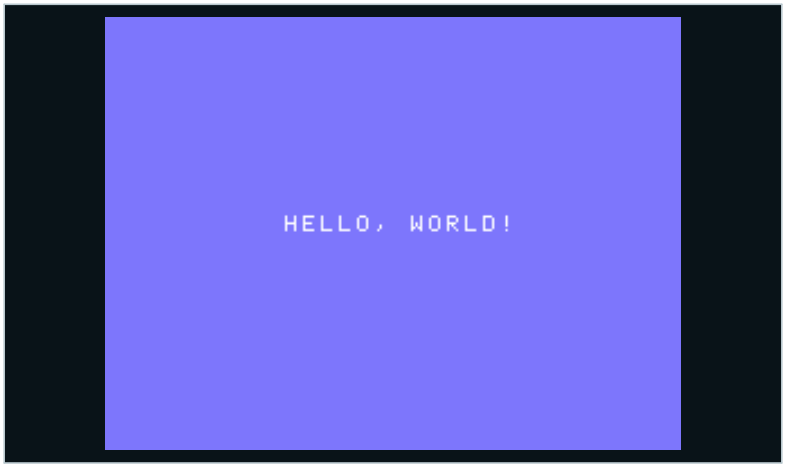
Occupied ROM size excludes cartridge padding. Lower is smaller, not automatically faster or better.

Amy Studio source and GearColeco results

Listings are complete except for bulky sprite patterns and the state-update engine body; both complete sources are in the evidence archive. Screenshots come from GearColeco running each measured ROM for 180 NTSC frames; current integrated debugging uses GearColeco 1.7.0.

Hello World

```
text screen
backdrop sky blue
print centered at 11, "HELLO, WORLD!"
screen on
```



GearColeco screenshot

GearColeco output after 180 NTSC frames

Warrior Graphics II Bitmap

```
picture WarriorPicture:
  pattern from "@project/warrior.pattern.zx0" codec zx0
  color from "@project/warrior.color.zx0" codec zx0
end picture

show picture WarriorPicture
```



GearColeco screenshot

Exact GearColeco output; all eight tools reproduce the same target image

Visual Controller Input

```
text screen
backdrop dark blue
print centered at 8, "CONTROLLER TEST"
print centered at 11, "MOVE OR PRESS A BUTTON"
print centered at 14, "KEYPAD IS PURPLE"
screen on

InputLoop:
  wait
  if keypad(1) <> 255 then
    backdrop magenta
  elseif joypad(1).fire then
    backdrop light red
  elseif joypad(1).up then
    backdrop light green
  elseif joypad(1).down then
    backdrop dark green
  elseif joypad(1).left then
    backdrop light yellow
  elseif joypad(1).right then
    backdrop cyan
  else
    backdrop dark blue
  end if
  goto InputLoop
```



GearColeco screenshot

GearColeco output at the neutral controller state

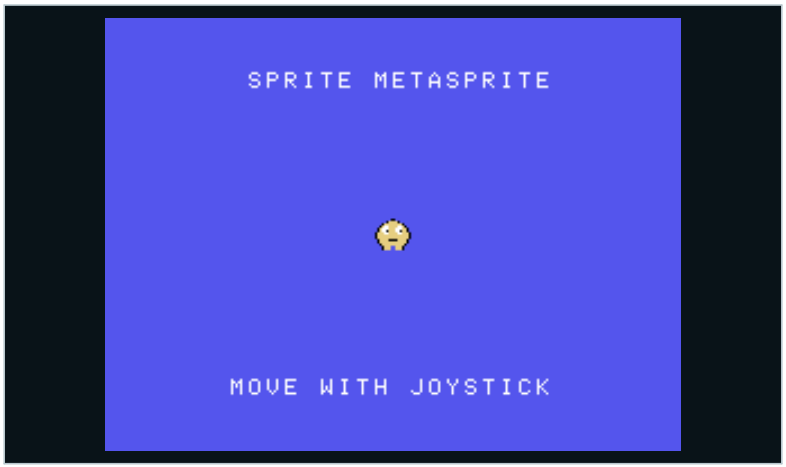
Animated Three-Color Metasprite

```
u8 PlayerX = 120
u8 PlayerY = 88
u8 AnimationTick = 0
u8 AnimationFrame = 0

data PlayerMeta metasprite layers 3
  frame 0,15, 4,11, 8,1
  frame 12,15, 16,11, 20,1
end data

' Six 16x16 pattern blocks are included in the evidence archive.
sprites simple
set sprite count 3

MainLoop:
  wait
  if joypad(1).left and PlayerX > 1 then PlayerX -= 1
  if joypad(1).right and PlayerX < 239 then PlayerX += 1
  if joypad(1).up and PlayerY > 1 then PlayerY -= 1
  if joypad(1).down and PlayerY < 175 then PlayerY += 1
  AnimationTick += 1
  if AnimationTick = 8 then
    AnimationTick = 0
    AnimationFrame = 1 - AnimationFrame
  end if
  set metasprite PlayerMeta frame AnimationFrame to PlayerY,PlayerX using sprite 0
  update sprites
  goto MainLoop
```



GearColeco screenshot

A metasprite is one visual actor assembled here from three overlapping hardware sprites

Gameplay State Update

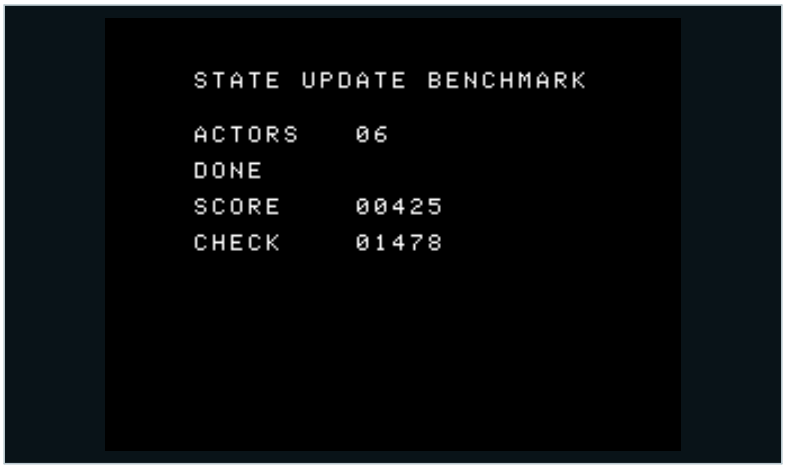
```
record ActorState:
  u8 X
  u8 Y
  u8 Direction
  u8 Timer
  u8 Mode
  u8 Hits
end record

state machine WorldBehavior:
  Running calls UpdateActors
  Bonus calls UpdateBonus
end state machine

ActorState Actors[6]
u16 Score = 0
u16 Checksum = 0

for Tick = 0 to 47
  dispatch WorldMode using WorldBehavior
next

sub UpdateActors:
  for each Actor, ActorIndex in Actors
    ' Movement, timers, modes, collision and score updates.
  next Actor
  if Tick = 23 then WorldMode = WorldBehavior.Bonus
  return
```



GearColeco screenshot

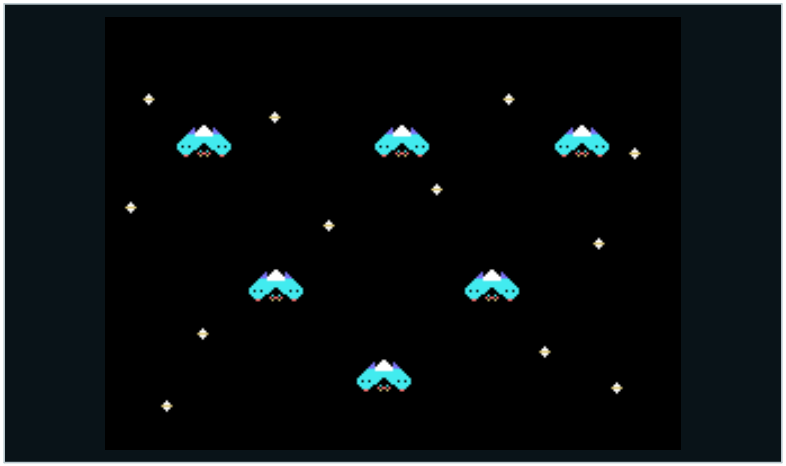
Experimental: 1,156 occupied bytes; exact oracle: state 1, collisions 13, score 425, checksum 1478

Graphics II Tile Animation

```
const StarTile = $80
const ShipFirstTile = $90
u8 StarFrame = 0
u8 ShipFrame = 0

tile screen
backdrop black
nmi off
' Install shared star and two 3x2 ship frames in all Graphics II thirds.
copy StarPattern0 count 8 to vram.pattern + StarTile * 8
copy ShipPatterns count 96 to vram.pattern + ShipFirstTile * 8
copy ShipColors count 96 to vram.color + ShipFirstTile * 8
nmi on

AnimationLoop:
  wait
  Tick += 1
  if Tick & 3 = 0 then
    ' Replace one shared pattern: every star animates together.
    StarFrame = (StarFrame + 1) % 3
    copy StarPatterns[StarFrame * 8] count 8 to vram.pattern + StarTile * 8
  end if
  if Tick & 7 = 0 then
    ' Replace six 3x2 Name Table frames without moving the ships.
    ShipFrame = 1 - ShipFrame
    for Ship = 0 to 5
      put ShipTiles[ShipFrame * 6] frame size 3,2 at ShipX[Ship],ShipY[Ship]
    next
  end if
goto AnimationLoop
```



GearColeco screenshot

Experimental: 1,316 occupied bytes; continuous shared-pattern and six-ship Name Table animation verified

Pure Z80 assembly reference

Amy Bienvenu published this BIOS-based Hello World on AtariAge in 2010 under the name Daniel Bienvenu. Amy's assembler builds it without optimization, and GearColeco verifies the exact VRAM text after BIOS startup. It occupies 184 bytes.

```
; COLECOVISION - HELLO WORLD!
; By Amy Bienvenu, published as Daniel Bienvenu in 2010
; Original AtariAge sample for TNIASM, assembled here by Amy's Z80 assembler.
CALC_OFFSET     equ $08c0
LOAD_ASCII      equ $1f7f
FILL_VRAM       equ $1f82
MODE_1          equ $1f85
PUT_VRAM        equ $1fbe
TURN_OFF_SOUND  equ $1fd6
WRITE_REGISTER  equ $1fd9
READ_REGISTER   equ $1fdc
WRITE_VRAM      equ $1fdf
VRAM_NAME       equ $1800
VRAM_COLOR      equ $2000
cpu Z80
org $8000
db $aa,$55
dw 0,0,0,0
dw Start
rst_8:  reti
        nop
rst_10: reti
        nop
rst_18: reti
        nop
rst_20: reti
        nop
rst_28: reti
        nop
rst_30: reti
        nop
rst_38: reti
        nop
jp Nmi
db "HELLO WORLD!/PRINT ON SCREEN/2010"
Start:
    im 1
    ld hl,$0000
    ld de,$4000
    xor a
    call FILL_VRAM
    call MODE_1
    call TURN_OFF_SOUND
```



Pure Z80 assembly reference - continued

```
    call LOAD_ASCII
    ld hl,VRAM_COLOR
    ld de,32
    ld a,$f0
    call FILL_VRAM
    call Static_Offset
    call Write_HelloWorld
    ld bc,$01c2
    call WRITE_REGISTER
TheEnd:
    jp TheEnd
HelloWorld:
    db "HELLO WORLD!"
Static_Offset:
    ld de,VRAM_NAME+10
    ret
; Alternative shown in the original tutorial: compute row 0, column 10.
Calculated_Offset:
    ld d,0
    ld e,10
    call CALC_OFFSET
    ld hl,VRAM_NAME
    add hl,de
    ex de,hl
    ret
Write_HelloWorld:
    ld hl,HelloWorld
    ld bc,12
    jp WRITE_VRAM
; Alternative BIOS routine shown in the original tutorial.
Put_HelloWorld:
    ld a,2
    ld hl,HelloWorld
    ld iy,12
    jp PUT_VRAM
Nmi:
    ld hl,VRAM_COLOR
    ld de,32
    ld a,r
    and $f0
    call FILL_VRAM
    call READ_REGISTER
    retn
```

Bienvenu also authored NewColeco. As the AtariAge discussion noted, this compact listing reuses the ColecoVision BIOS font and VDP routines; a private font and runtime would be more independent but larger. Matching hand-written ASM versions now cover all six fixtures and pass their GearColeco oracles. Together they occupy 5,126 bytes versus 7,372 for Amy Studio, a 2,246-byte difference. The current build includes VDP-critical NMI deferral and a bounded frame-wait fallback; the latter prevents a reproducible missed-NMI stall.

The controller ROM fell from 297 to 264 bytes by replacing CONT_SCAN plus a 16-byte keypad table with two BIOS DECODER calls and by storing only the final display color. POLLER remains appropriate when two-sample debounce is required, but its 12-byte result contract and 83-91 executed BIOS instructions per call do not improve this fixture; one DECODER segment takes 29 instructions.

Graphics II compression corpus

All 42 unique 12,288-byte pictures passed exact round-trip checks with every codec. The gallery shows the actual TMS9918 output, the three smallest payloads, and Pletter's rank when it falls outside the top three. Exomizer also needs a 226-byte ROM decoder and 156 bytes of CPU RAM. Amy Studio reserves that workspace at a 256-byte boundary; its generated address varies and must not overlap other RAM.

Rank	Codec	Payload total	Decoder	First-use places (pictures)				
				1st	2nd	3rd	4th	5th
1	Exomizer 2	161,374	226	21	12	2	3	1
2	ZX0	166,556	136	21	18	1	2	0
3	DAN2	166,923	212	0	0	7	13	10
4	DAN3	167,186	205	0	0	16	5	11
5	DAN1	167,240	205	0	3	4	9	16
6	MegaLZ	171,016	162	0	0	1	1	1
7	aPLib Compact	171,484	244	0	0	0	0	0
8	Pletter	171,751	212	0	0	0	0	0
9	ZX7	171,788	136	0	0	2	9	3
10	BitBuster	172,941	166	0	0	0	0	0
11	ZX1	176,562	127	0	4	5	0	0
12	ZX2	177,582	115	0	5	4	0	0
13	LZF	193,412	117	0	0	0	0	0
14	Nibble	214,555	115	0	0	0	0	0
15	MDK-RLE	253,859	46	0	0	0	0	0
16	MSC1	259,023	68	0	0	0	0	0

Corpus gallery 1-6



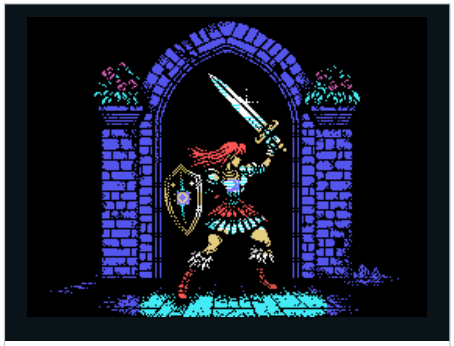
Cake
Amy Bienvenu / NewColeco
1. Exomizer 2: 2,935 bytes
2. ZX0: 2,958 bytes
3. DAN2: 2,999 bytes
... 9. Pletter: 3,108 bytes



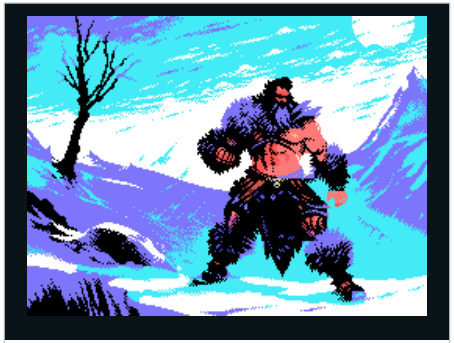


Commando
Amy Bienvenu / NewColeco
1. Exomizer 2: 5,206 bytes
2. ZX0: 5,310 bytes
3. DAN2: 5,312 bytes
... 8. Pletter: 5,508 bytes





Warrior
1. Exomizer 2: 2,821 bytes
2. ZX0: 2,849 bytes
3. DAN3: 2,891 bytes
... 10. Pletter: 2,988 bytes



Barbarian
1. Exomizer 2: 4,165 bytes
2. ZX0: 4,229 bytes
3. DAN3: 4,286 bytes
... 5. Pletter: 4,339 bytes



421 Title
Amy Bienvenu / NewColeco
1. ZX0: 1,055 bytes
2. Exomizer 2: 1,060 bytes
3. DAN2: 1,067 bytes
... 8. Pletter: 1,110 bytes

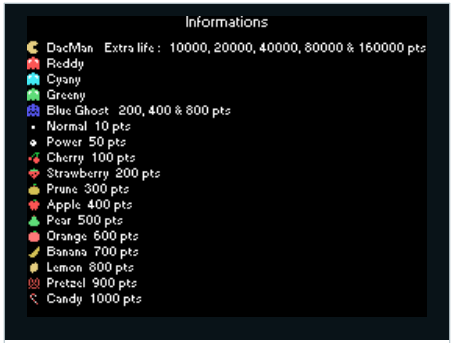




Dacman Title
Amy Bienvenu / NewColeco
1. Exomizer 2: 967 bytes
2. DAN1: 969 bytes
3. DAN2: 972 bytes
... 9. Pletter: 1,035 bytes



Corpus gallery 7-12



Dacman Information
Amy Bienvenu / NewColeco
1. Exomizer 2: 1,248 bytes
2. ZX0: 1,260 bytes
3. DAN2: 1,273 bytes
... 10. Pletter: 1,361 bytes





Dacman 2 Title
Amy Bienvenu / NewColeco
1. Exomizer 2: 1,142 bytes
2. DAN2: 1,168 bytes
3. DAN1: 1,171 bytes
... 8. Pletter: 1,231 bytes



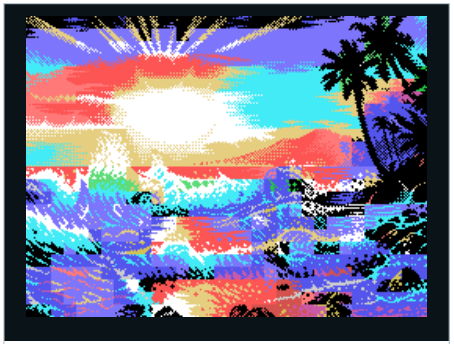


Le Chateau du Dragon
Amy Bienvenu / NewColeco
1. Exomizer 2: 1,613 bytes
2. ZX0: 1,636 bytes
3. DAN1: 1,651 bytes
... 10. Pletter: 1,730 bytes

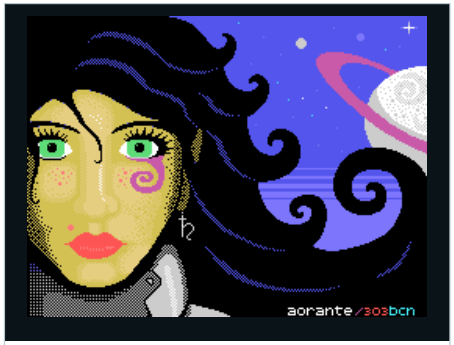




Almighty God - Aliquis Acuario Ingens
1. Exomizer 2: 7,099 bytes
2. ZX0: 7,234 bytes
3. DAN3: 7,271 bytes
... 7. Pletter: 7,447 bytes



Aloha (ArtField 2010, 1)
1. Exomizer 2: 6,754 bytes
2. DAN1: 6,989 bytes
3. DAN2: 6,991 bytes
... 9. Pletter: 7,199 bytes



Aorante - Lady Saturn
1. Exomizer 2: 3,497 bytes
2. DAN3: 3,653 bytes
3. DAN2: 3,657 bytes
... 8. Pletter: 3,810 bytes

Corpus gallery 13-18



Bejeweled
Amy Bienvenu / NewColeco
1. Exomizer 2: 1,519 bytes
2. ZX0: 1,564 bytes
3. DAN1: 1,594 bytes
... 9. Pletter: 1,661 bytes



Anime Portrait
1. Exomizer 2: 2,098 bytes
2. DAN2: 2,157 bytes
3. DAN3: 2,162 bytes
... 8. Pletter: 2,265 bytes



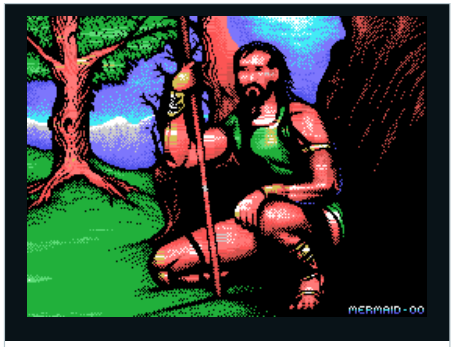
Band Wagon Logo
Vanja Utne / Mermaid
1. Exomizer 2: 2,155 bytes
2. ZX0: 2,172 bytes
3. DAN2: 2,183 bytes
... 6. Pletter: 2,243 bytes



Bat Character
Vanja Utne / Mermaid
1. Exomizer 2: 945 bytes
2. ZX0: 980 bytes
3. DAN1: 987 bytes
... 8. Pletter: 1,051 bytes



Blue Portrait
Vanja Utne / Mermaid
1. Exomizer 2: 5,902 bytes
2. ZX0: 6,073 bytes
3. DAN2: 6,097 bytes
... 7. Pletter: 6,224 bytes

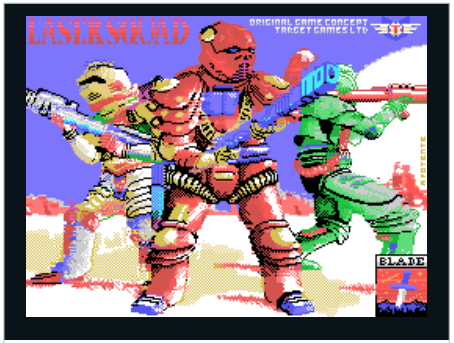


Fantasy Camp
Vanja Utne / Mermaid
1. Exomizer 2: 5,996 bytes
2. DAN2: 6,138 bytes
3. DAN1: 6,147 bytes
... 8. Pletter: 6,317 bytes

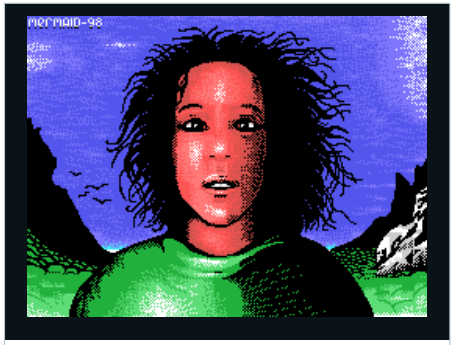
Corpus gallery 19-24



Feliz Navidad
1. Exomizer 2: 3,998 bytes
2. DAN3: 4,149 bytes
3. ZX0: 4,154 bytes
... 9. Pletter: 4,300 bytes



Laser Squad
1. Exomizer 2: 7,357 bytes
2. DAN2: 7,488 bytes
3. DAN1: 7,494 bytes
... 8. Pletter: 7,774 bytes

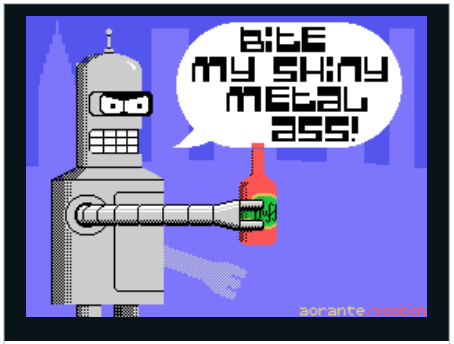


Mermaid Portrait
Vanja Utne / Mermaid
1. Exomizer 2: 5,848 bytes
2. ZX0: 6,050 bytes
3. DAN2: 6,070 bytes
... 7. Pletter: 6,196 bytes

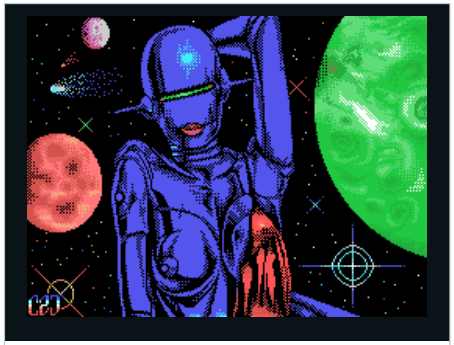




Reservoir Dogs
1. Exomizer 2: 2,208 bytes
2. ZX0: 2,315 bytes
3. DAN3: 2,316 bytes
... 8. Pletter: 2,414 bytes



Robot Quote
1. Exomizer 2: 1,990 bytes
2. DAN3: 2,076 bytes
3. DAN2: 2,097 bytes
... 9. Pletter: 2,217 bytes



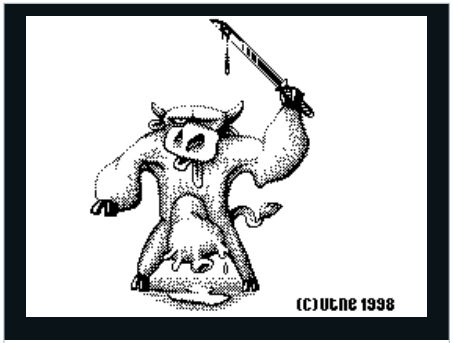
Space Soldier
1. Exomizer 2: 5,522 bytes
2. DAN2: 5,649 bytes
3. DAN1: 5,661 bytes
... 9. Pletter: 5,886 bytes

Corpus gallery 25-30



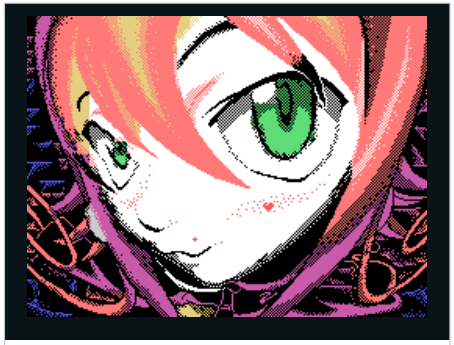
Turban Character
Vanja Utne / Mermaid
1. Exomizer 2: 7,629 bytes
2. DAN1: 7,908 bytes
3. DAN2: 7,911 bytes
... 7. Pletter: 8,170 bytes





Mad Cow
Vanja Utne / Mermaid
1. Exomizer 2: 2,125 bytes
2. DAN3: 2,171 bytes
3. DAN2: 2,180 bytes
... 9. Pletter: 2,273 bytes

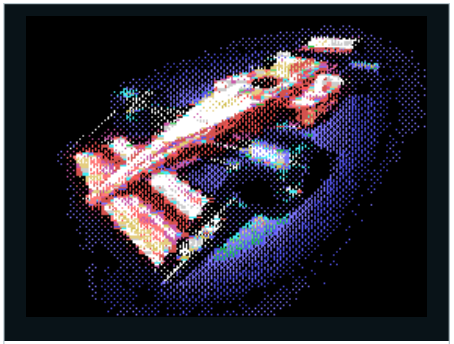




Blair (Chaos Constructions 2009, 6)
1. Exomizer 2: 5,140 bytes
2. ZX0: 5,372 bytes
3. DAN3: 5,374 bytes
... 8. Pletter: 5,491 bytes



Bugs (Demobit 1995, 4)
1. Exomizer 2: 2,032 bytes
2. DAN3: 2,091 bytes
3. ZX0: 2,107 bytes
... 9. Pletter: 2,191 bytes

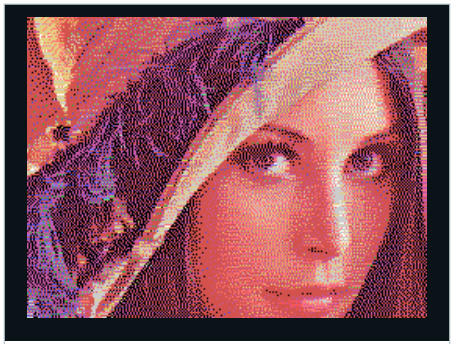


F1 SPP
1. Exomizer 2: 4,169 bytes
2. DAN3: 4,277 bytes
3. DAN2: 4,278 bytes
... 9. Pletter: 4,462 bytes



Kozmonaut (Forever 7, 7)
1. Exomizer 2: 2,694 bytes
2. DAN3: 2,780 bytes
3. ZX0: 2,792 bytes
... 9. Pletter: 2,892 bytes

Corpus gallery 31-36



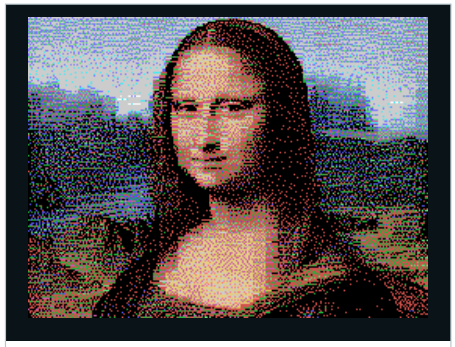
lenna
1. Exomizer 2: 9,071 bytes
2. ZX0: 9,400 bytes
3. DAN2: 9,425 bytes
... 8. Pletter: 9,556 bytes



Maze Maniac
Amy Bienvenu / NewColeco (contributor)



1. Exomizer 2: 2,625 bytes
2. DAN3: 2,709 bytes
3. ZX0: 2,713 bytes
... 7. Pletter: 2,817 bytes



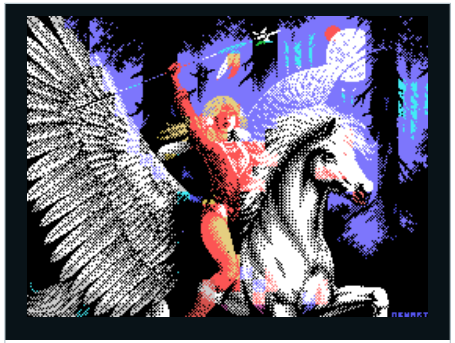
Mona Lisa
1. Exomizer 2: 9,208 bytes
2. ZX0: 9,426 bytes
3. DAN3: 9,490 bytes
... 5. Pletter: 9,640 bytes



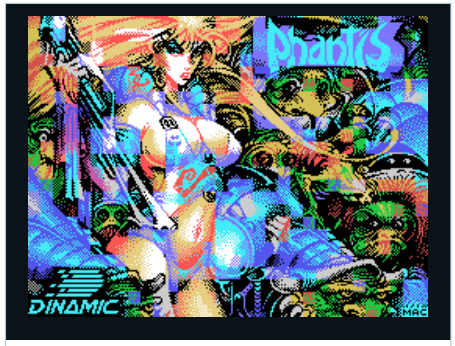
ROM File Edition
Amy Bienvenu / NewColeco



1. ZX0: 901 bytes
2. Exomizer 2: 904 bytes
3. DAN2: 909 bytes
... 9. Pletter: 960 bytes



Pegasus (Millennium 1903, 1)
1. Exomizer 2: 6,168 bytes
2. ZX0: 6,405 bytes
3. DAN1: 6,415 bytes
... 8. Pletter: 6,544 bytes



Phantis (3BM OpenAir 2014, 1)
1. Exomizer 2: 8,109 bytes
2. DAN1: 8,387 bytes
3. DAN2: 8,388 bytes
... 8. Pletter: 8,599 bytes

Corpus gallery 37-42

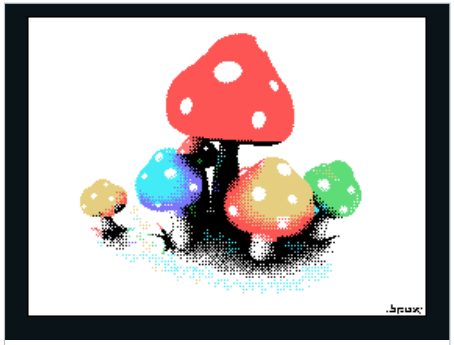


Robee Blaster Title
Alek Maul / AlekMaul
1. Exomizer 2: 2,024 bytes
2. ZX0: 2,082 bytes
3. DAN2: 2,104 bytes
... 8. Pletter: 2,138 bytes

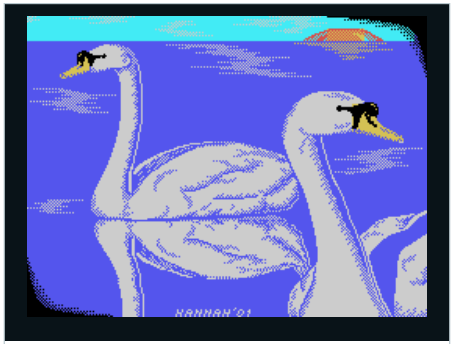




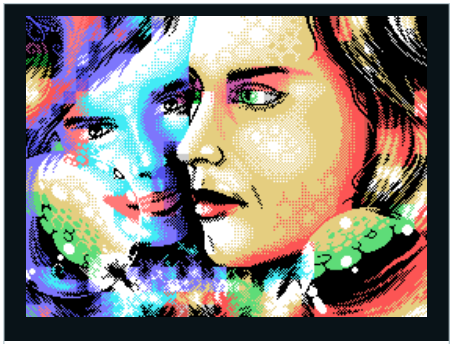
Smurf Challenge
Jean-Philippe Meola / Youki
1. Exomizer 2: 1,227 bytes
2. DAN1: 1,264 bytes
3. DAN2: 1,269 bytes
... 8. Pletter: 1,342 bytes



Stainly Rainbow (ArtField 2006, 3)
1. Exomizer 2: 2,170 bytes
2. ZX0: 2,248 bytes
3. DAN2: 2,253 bytes
... 8. Pletter: 2,358 bytes



Vranov (Chaos Constructions 2001, 7)
1. Exomizer 2: 3,396 bytes
2. ZX0: 3,475 bytes
3. DAN3: 3,494 bytes
... 8. Pletter: 3,599 bytes



Xzema (Chaos Constructions 2001, 2)
1. Exomizer 2: 6,544 bytes
2. DAN1: 6,798 bytes
3. DAN2: 6,815 bytes
... 9. Pletter: 7,079 bytes



Znachok (Enlight 1996, 15)
1. Exomizer 2: 2,094 bytes
2. DAN3: 2,134 bytes
3. ZX0: 2,151 bytes
... 6. Pletter: 2,226 bytes

Updated: 2026-11-06

Method

This comparison separates four kinds of evidence:

- **Runtime verified**: compiled for ColecoVision and checked in an emulator or self-test.
- **Build verified**: the ColecoVision source, header, library, or generated assembly was inspected.
- **Documented**: claimed by the tool's official documentation, but not yet measured here.
- **Unverified**: available somewhere in the wider tool, but not proven for ColecoVision.

Amy Studio combines a language and IDE; the others are mainly compilers or C kits. IDE-only features are listed separately from language capabilities.

The eight toolchains and ASM baseline

Solution	Primary approach	ColecoVision position
Amy Studio	ColecoVision-first Amy language and browser IDE	Stock 1 KB RAM cartridge target, with optional hardware simulation in the debugger
CVBasic	Portable integer BASIC cross-compiler	Mature native target; optional MegaCart and SGM support
z88dk `+coleco`	General Z80 C toolchain	C compiler, assembler, linker, libraries, and app packaging
ugBASIC	Portable retro BASIC compiler	Declared target with common high-level multimedia vocabulary
devkitSMS + `SGlib_CV`	SDCC C plus compact game libraries	ColecoVision adaptation of the SG/SMS development workflow
PVColLib	ColecoVision-focused SDCC C library and devkit	Native VDP, controller, sprite, sound, music, and compression APIs
NewColeco	Historical SDCC C, `CRTCV`, `CVLIB`, and GETPUT 1.1	Amy's pre-Studio ColecoVision workflow and direct ancestor of current techniques
PkK's devkit	Philipp Klaus Krause's ColecoVision SDCC C devkit, using the `libcv` / `libcvu` libraries	Small link-only runtime with native graphics, input, sprites, and RLE+Huffman assets

Reproducible six-sample ROM suite

The suite builds six runnable programs with all eight toolchains, plus matched Pure ASM baselines:

1. a visible Hello World;
2. the Warrior Graphics II bitmap picture;
3. a visual controller monitor;
4. an animated, controller-driven three-color metasprite;
5. a deterministic six-actor gameplay state update;
6. a Graphics II star field and six animated 3x2 tile ships.

A **metasprite** combines hardware sprites into one actor. This test overlaps three 16x16 layers (white, yellow, black), below the four-sprites-per-scanline limit.

The current evidence package contains 62 complete, runtime-verified ROMs, grouped under `build/competition` and packaged with their sources and measurements. `tools/report-five-tool-sample-sizes.ps1` records occupied sizes; dedicated GearColeco tests verify the bitmap, controller, metasprite, and deterministic state-update oracles.

Validated optimization settings

Solution	Setting used	Observation
Amy Studio	Experimental	All six runtime oracles pass; current sizes include VDP-safe NMI deferral
CVBasic `main` `e84bf978` (2026-11-05)	Normal compiler; TMSColor `-z -p2` for Bitmap	All six rebuilt and runtime-validated; Pletter is used for its bitmap
z88dk	zsdcc size profile; direct `cputs`; classic backend + ZX0 for Bitmap	Coleco-safe direct-to-VRAM port of z88dk's ZX0 core; its classic ABI is retained for this hand-written depacker
ugBASIC `main` `2db87ebd` (2026-11-01)	Default maximum of 16 peephole passes; `DEFINE DEFAULT TYPE BYTE`; `PRINT RAW`; MSC1 bitmap compression	Latest source reduces all six samples; bitmap, controller, sprite, state and tile-animation oracles pass
devkitSMS / SDCC 4.6	`--opt-code-size`; best validated per-sample node budget from 100,000 or 500,000	All six rebuilt with corrected ColecoVision startup and runtime oracles
PVColLib 1.6.0 / bundled SDCC	`--opt-code-size`; best validated per-sample node budget from 5,000 through 500,000	Official ABI retained; linked only referenced library modules
NewColeco / SDCC 2.9.0	`--std-c99`; original prebuilt libraries plus historical DAN2	Historical stack ABI required; the exact DAN2 bitmap is 626 bytes smaller than its GETPUT MDKRLE baseline
PkK's devkit / SDCC 4.5	`--opt-code-size`; best validated per-sample node budget from 10,000 through 500,000	Its `libcv` / `libcvu` libraries link only referenced modules; incompatible modern-SDCC graphical builds were rejected

These are the most size-oriented **native settings validated in this study**, not claims about every release or project configuration. No other toolchain passed through Amy's optimizer or MDL. Experimental serves this size test; Balanced remains Amy's default.

SDCC allocator searches are non-monotonic, so devkitSMS, PVColLib, PkK/libcv and z88dk use the smallest runtime-valid result per sample rather than one arbitrary global node budget. Builds that were smaller but changed VRAM, framebuffer, state, animation, or ABI behavior were rejected. The PVColLib and NewColeco controller fixtures set transparent glyph backgrounds (`$F0`) so the same changing VDP backdrop remains visible; this presentation-only source normalization changes neither controller logic nor occupied size. ugBASIC's `DEFINE BIOS ALLOWED` remains separately documented but unranked until the same current compiler can build and pass both variants locally.

ugBASIC source settings were re-audited with current `main` commit `2db87ebd`. All six programs use `DEFINE DEFAULT TYPE BYTE`; text fixtures use `PRINT RAW`; Controller tests its byte-valued keypad result directly with `IF k`; Warrior uses native `EXACT COMPRESSED` MSC1 data. The selected build occupied 20,022 bytes on the original six fixtures, but the corrected metasprite oracle rejected that revision's 3,295-byte ROM for incoherent frame updates. Replacing six independently managed sprites with ugBASIC's native `CSPRITE` grouping makes every rendered frame coherent and reduces the row to 2,883 bytes. The corrected six-sample total is 19,610 bytes. All published ROMs are retained in the evidence archive with SHA-256 hashes.

Real occupied size, excluding cartridge padding

Sample	Pure ASM*	Amy Studio	NewColeco	PkK's devkit	PVColLib	CVBasic	devkitSMS	z88dk	ugBASIC
Hello World	**184**	234	795	1,001	1,106	816	1,386	1,980	1,340
Warrior bitmap	**3,066**	3,213	3,643	4,682	4,525	4,313	4,665	4,976	7,580
Controller Visual	**264**	604	932	812	1,194	1,086	1,388	2,272	1,917
Sprite Metasprite	**482**	797	1,166	1,413	1,201	1,261	1,637	2,492	2,883
Gameplay State Update*	**508**	1,141	1,253	1,225	1,229	1,890	2,023	2,672	3,931
Tile Animation	**761**	1,255	1,376	1,627	1,516	2,332	1,822	2,729	1,959
Six-sample total*	**5,265**	**7,244**	**9,165**	**10,760**	**10,771**	**11,698**	**12,921**	**17,121**	**19,610**

***** Pure ASM is a lower-level diagnostic baseline; its State Update omits Amy Studio's status display.

Measured bitmap baselines: z88dk RAW 14,293 bytes, MDKRLE 5,911, ZX7 5,115, and ZX0 4,976; NewColeco GETPUT/MDKRLE 4,269 and DAN2 3,643. All reproduce both VRAM tables and 49,152 pixels. The 7,580-byte ugBASIC Warrior uses native **`BITMAP ENABLE(16)`**, **`CLS BLACK`**, **`LOAD IMAGE ... EXACT COMPRESSED`**, and **`PUT IMAGE`**. The oracle finds 450 Pattern-byte and 1,322 Color-byte differences caused by equivalent foreground/background encodings, while all 49,152 rendered pixels match. The MSC1 direct-to-VRAM result is pixel-exact.

Legacy payloads are MDKRLE 3,687, DAN1 2,903, DAN2 2,897, and DAN3 2,891 bytes. DAN2 stays linked: DAN3 saves six payload bytes before decoder cost, while DAN2 is validated end to end.

`Occupied` includes header, runtime, code, and data, but excludes cartridge padding. Sources are: assembled length (Amy), **`ROM_END-$8000`** (CVBasic), unpadded binary (z88dk), generated **`code_user`** ROM section (ugBASIC), Intel HEX span above **`$8000`** (devkitSMS/PVColLib), and complete linked binary (NewColeco).

ugBASIC **`main 2db87ebd`** (2026-11-01) is the measured revision for all six samples. Its original metasprite ROM fails the corrected dynamic oracle because sprite layers can come from different frames. The revised source uses native **`CSPRITE`** groups and occupies 2,883 bytes; the oracle compares final rendered pixels so equivalent covered-mask removal is accepted without weakening the coherent- frame requirement. Occupied size uses the linker's exclusive **`__code_user_tail`** where available; reserved RAM and cartridge padding remain excluded.

Runtime and comparability verdict

Sample	Runtime result	Verdict
Hello World	Nine ROMs complete 180 GearColeco frames	Stable startup
Warrior bitmap	Nine pipelines reproduce the 256x192 image	All ranked framebuffers are pixel-exact; equivalent PATTERN/COLOR encodings are allowed
Controller Visual	Nine pass injected neutral, keypad, UP, FIRE, and release states	ugBASIC uses Coleco direction bits `$01/$02/$04/$08` and FIRE bit `$40`
Sprite Metasprite	Nine pass the same controller-driven dynamic oracle; ugBASIC uses native `CSPRITE` groups	Movement, bounds, animation, and coherent rendered layers, colors, and priority
Gameplay State Update	Nine match world state 1, 13 collisions, score 425, checksum 1478	Exact deterministic oracle
Tile Animation	Nine continuously animate shared patterns and six 3x2 Name Table frames	Exact VRAM tables, no stray ship tiles, animation continues after 100 frames

The State Update row is behavior-validated but not yet a strict code-size comparison: Amy Studio also draws a title, progress text, score, and checksum. The NewColeco, PVColLib, and PkK/libcv fixtures share the same C engine and run it without equivalent display output. Their sizes remain evidence artifacts, not a code-generator ranking, until matched headless and visible variants are built. Amy's verified headless engine is 817 bytes, smaller than NewColeco (1,253), PkK/libcv (1,291), and PVColLib (1,308).

PVColLib and NewColeco use `$F0` transparent-background text so VDP R7 changes remain visible; controller logic and occupied size are unchanged.

BIOS-assisted pure Z80 assembly baseline

Amy Bienvenu published this TNIASM-style `HELLO WORLD!` tutorial on AtariAge in 2010 under the name Daniel Bienvenu; she is also the author of NewColeco. Amy's assembler builds it without optimization, and GearColeco verifies the exact twelve Name Table bytes after BIOS startup. The program occupies **184 bytes**, measured by `tools/test-pure-asm-hello.mjs`.

The historical listing is compact because it reuses the BIOS font and VDP routines. Matching hand-written ASM implementations now exist for all six fixtures, use the same assets and observable behavior allowed to Amy Studio, and pass their GearColeco runtime oracles.

Sample	Pure ASM	Amy Studio	ASM difference
Hello World	184	234	-50
Warrior bitmap	3,066	3,213	-147
Controller Visual	264	604	-340
Sprite Metasprite	482	797	-315
Gameplay State Update	508	1,141	-633
Tile Animation	761	1,255	-494
Six-sample total	**5,265**	**7,244**	**-1,979**

The controller fixture originally occupied 297 bytes using BIOS `CONT_SCAN`, cartridge-side decoding, and a 16-byte keypad table. The optimized 264-byte version calls BIOS `DECODER` once for joystick/left FIRE and once for keypad/right FIRE, removes the lookup table, and stores only the final display color. It passes neutral, keypad, held direction, FIRE, and release tests. BIOS `POLLER` remains useful when its two-sample debounce is required, but its 12-byte result contract and 83-91 executed BIOS instructions per call are not a free size optimization; one `DECODER` segment takes 29 instructions.

The ASM total is a lower-level baseline, not evidence that assembly is universally preferable. It measures the price of language/runtime convenience and identifies concrete optimization opportunities. The State Update caveat still applies: its ASM fixture validates the complete headless state workload but does not reproduce Amy's extra status display.

Native bitmap paths are Amy/z88dk ZX0, CVBasic Pletter, ugBASIC MSC1 resources, devkitSMS aPLib, PVColLib RLE, PkK's devkit using `libcv` RLE+Huffman, and NewColeco DAN2. All framebuffers are exact despite equivalent internal table encodings. PVColLib Pletter failed VRAM validation and is excluded.

For z88dk, MDKRLE reaches VRAM at frame 93, ZX0 at 132, and ZX7 at 138. ZX0 saves 935 bytes versus MDKRLE and 139 versus ZX7; MDKRLE is faster. A transplanted SMS aPLib path failed and is excluded; native devkitSMS aPLib remains exact with frame interrupts disabled during decompression.

The target-specific ugBASIC RGB palette removes avoidable nearest-color errors. Its current full-screen Warrior framebuffer is pixel-exact even though the internal tables use equivalent encodings.

Observations supported by this suite

- Amy produces the smallest measured ROM for Hello, Controller, and this exact bitmap fixture

(3,213 versus CVBasic's 4,313 bytes for the bitmap).

- Portable C/BASIC runtimes add visible fixed cost, but complete games may scale differently.
- Cartridge length includes packaging; a padded 32 KB file is not necessarily a 32 KB program.
- Smaller results count only after runtime validation; this finite suite cannot establish a

universal winner.

- RAM, worst-frame cycles, build latency, sound, sprite pressure, and gameplay remain separate tests.

Preliminary capability matrix

Legend: **Yes** = verified or explicit target support; **Partial** = manual or narrower support; **Pending** = must still be proven specifically on ColecoVision. ***** marks a benchmark adaptation written here, not a native ColecoVision direct-to-VRAM path supplied by that solution.

This compact matrix covers six current general-purpose toolchains. NewColeco and PkK's devkit are documented separately below from their measured fixtures and inspected APIs.

Capability	Amy Studio	CVBasic	z88dk	ugBASIC	devkitSMS / SGlib_CV	PVColLib
8/16-bit integers	Yes	Yes	Yes	Documented	Yes, through SDCC	Yes, through SDCC
32-bit integers	Yes, runtime tested	No native BASIC type	Yes	Documented; target cost pending	Yes, through SDCC	Yes, through SDCC
Fixed-point	Yes, 8.8 and 16.16	No	Library/manual	Documented; target proof pending	Manual C/library code	Manual C/library code
Records/structures	Yes	No	Yes	Documented custom types	Yes	Yes
2D arrays	Yes	No native 2D syntax	Yes	Documented	Yes	Yes
RAM overlays	Yes, first-class	Manual	C union/manual layout	Pending	C union/manual layout	C union/manual layout
Functions with values and returns	Yes	Limited BASIC procedures/DEF FN	Yes	Yes	Yes	Yes
Inline Z80 assembly	Yes	Yes	Yes	Documented	Yes	Yes
TMS9918 screen and tile operations	Yes	Yes	Yes	Yes, target proof incomplete	Yes	Yes
Hardware sprite API	Yes	Yes	Low-level/library	Yes, exact target surface pending	Yes	Yes, 32-entry SAT
Four-sprite scanline mitigation	Yes, stable ranges and flicker	Yes, sprite flicker	Manual	Pending	Manual/library-dependent	Yes, NMI sprite swapping
Keypad	Yes	Yes	Library/manual	Pending	Yes/target library	Yes, both ports
Spinner / Roller Controller	Yes	Yes	Not confirmed	Pending	Not confirmed	Yes, both ports
Held, pressed, and released input	Yes	Held; edges are manual	Manual	Manual edges over `JOY`	Yes, computed from NMI snapshots	Manual edges over NMI snapshots
Coleco PSG sound and voice	BIOS tables, Tiny Sound, DSOUND, VoxPCM, and SP0256 Lundy/EVE modules	Sound/music commands	Sound libraries	Sound commands, target proof pending	PSGlib_CV	BIOS-style sound tables and sequenced music
Direct-to-VRAM compression	Sixteen active codecs, including Exomizer 2 and MegaLZ	Pletter	RAM APIs; ZX0, *ZX1*, ZX2, *and ZX7* benchmark VRAM ports	Resource conversion; RAM-oriented compression	ZX7 and aPLib	RLE, Pletter, DAN1/2/3
ROM banking	Verified 64/128 KB MegaCart images and `bank select`; arbitrary banked Amy linking remains open	Yes	Yes	Pending	SMS workflow has banking; CV support pending	MegaCart tools and examples
Source-level Coleco debugger	Integrated	External emulator	External debugger/emulator	External or IDE-dependent	External debugger/emulator	External debugger/emulator
Rewind, breakpoints, VRAM/RAM inspection	Integrated	External	External	External	External	External
Graphics editors and project assets	Integrated bitmap, tile, sprite, frame, tilemap, and composite editors	Separate tools	Separate tools	Conversion-oriented IDE/tools	Separate asset tools	`gfx2col` and separate asset tools
Automated ROM behavior tests	Integrated project pipeline	Not supplied as one system	Can be assembled manually	Not established	Not established	Not established

Detailed language and runtime evidence

Data model

Area	Amy Studio	CVBasic	z88dk	ugBASIC	devkitSMS	PVCollib
Integer widths	Explicit signed/unsigned 8/16/32	8/16, signed or unsigned	C 8/16/32	Backend defines signed/unsigned 8/16/32	SDCC C 8/16/32	SDCC C 8/16/32
Fractional math	`fixed`, `ufixed`, `fixed32`, `fp5`	None	C/library float and fixed libraries	`VT_FLOAT` exists; Coleco cost pending	SDCC float/manual fixed	SDCC float/manual fixed
Decimal scores	Packed BCD 1-12 digits	Manual	Manual/library	Manual	Manual/library	Manual/library
Aggregates	Nested records, record arrays	None	struct/union	Typed arrays and internal resource types; user-structure surface pending	struct/union	struct/union
Multidimensional arrays	Primitive 2D arrays	One-dimensional only	Native C	Typed arrays; dimensional coverage pending	Native C	Native C
Mutually exclusive RAM	First-class overlays/scenes	Manual aliases	union/linker/manual	Banking/resource allocator; no equivalent proven	union/linker/manual	union/linker/manual
Dynamic strings	Deliberately fixed-buffer oriented	Mostly literals/printing	C strings and allocation libraries	First-class strings	C strings; risky in 1 KB RAM	C strings; risky in 1 KB RAM

Amy's data model directly addresses stock ColecoVision game RAM through overlays and scenes. C offers general-purpose unions and pointers but leaves lifetime analysis and debugger-aware overlay tracking to the project. ugBASIC has a broad internal type system; its ColecoVision ROM/RAM cost still requires fixture-specific compilation and validation.

Control, timing, and code organization

Area	Amy Studio	CVBasic	z88dk	ugBASIC	devkitSMS	PVCollib
Procedures/functions	Typed values, `ref`, returns, recursion	Procedures without value parameters; `DEF FN` expression macros	Full C	Procedures/functions documented	Full C	Full C
State machines	Typed zero-RAM dispatch	Manual `SELECT`/labels	Manual enum/switch/table	Manual control constructs	Manual enum/switch/table	Manual enum/switch/table
VBlank	`wait`, frame hooks, timers	`WAIT`, `ON FRAME`	HALT/NMI hooks/manual	`WAIT VBL`, `EVERY`, timers	`SG_waitForVBlank`, frame handler	`vdp_waitvblank`, user `nmi` hook
Parallel animation	Not yet first-class	Manual	Manual/interrupt	`ANIMATION`, `ANIMATE`, `MOVE`, paths and multitasking source present	Manual frame service	Manual NMI/frame service
Inline assembly	Amy ASM bridge	`ASM`, `CALL`, `USR`	Native inline/external ASM	Supported	SDCC inline/external ASM	SDCC inline/external ASM

ugBASIC provides a useful reference for a future optional Amy animation service. Its broader abstraction also illustrates costs Amy would need to measure: state variables, thread handles, paths, delays, signals, and background-preservation storage. Amy should evaluate those ideas only when exact RAM and cycle costs are visible and the entire service disappears when unused.

TMS9918 graphics and sprites

Area	Amy Studio	CVBasic	z88dk	ugBASIC	SGlib_CV	PVColLib
Text/tile/bitmap modes	Coleco-specific commands and editors	Modes and direct VRAM commands	Generic TMS9918/graphics libraries	TMS9918 backend and resource conversion	Tile, bitmap, VDP and VRAM functions	Text, bitmap, VDP and VRAM functions
Pixel/line/circle	Yes, programmer controls NMI-safe batching	Yes	Generic graphics	TMS9918 drawing backend	Pixel/bitmap helpers	Low-level VRAM/bitmap support
Sprites	Fields, clipping helpers, movement, editor	Define/place sprites	Low-level/library	Sprite conversion and TMS9918 operations	SAT builder, clipping, finalize/copy	32-entry SAT, update and fast-update APIs
Collision	Hitboxes, tile groups, hardware status	VDP status/manual	Hardware/manual	Hardware collision/hit backend	VDP status/manual	`spr_collide` hitboxes and VDP status
Scanline overflow	Explicit flicker and stable priority ranges	Sprite flicker	Manual	No verified policy	Manual ordering	`spr_getentry` swaps sprites each NMI
Asset authoring	Integrated bitmap/tile/sprite/frame editors	Separate tools	Separate tools	Automatic image conversion	External asset tools	`gfx2col` and external tools

Amy's lead is the complete authoring/debugging workflow and explicit TMS9918 policy. ugBASIC's automatic modern-image conversion is substantial, while SGlib_CV offers a compact C SAT workflow.

Controllers

Capability	Amy Studio	CVBasic	z88dk	ugBASIC	SGlib_CV	PVColLib
Two directions/fire ports	Yes	Yes	`joystick(1/2)`	`JOY(0/1)` backend	NMI snapshots both ports	`joypad_1/2`, four fire bits
Two keypad ports	Yes	Yes	`joystick(3/4)` high byte	`SCANCODE` path; exact port behavior pending	`SG_readCVNumPad`	`keypad_1/2`
Pressed/released	First-class per property	Manual previous state	Manual previous state	Manual previous state	First-class status helpers	Manual previous-state comparison
Spinner/Roller	Yes	Yes	Not found in Coleco target	Not found yet	Not found	`spinner_1/2`, reset and enable APIs
Automatic minimal backend	Yes, capability selected	Runtime-integrated	Linker includes referenced routines	Deployment system includes used modules	Library/NMI input service	Linker extracts referenced modules

Sound and music

Area	Amy Studio	CVBasic	z88dk	ugBASIC	devkitSMS	PVColLib
SN76489 tones/noise	BIOS sound tables and direct formats	`SOUND`	PSG libraries/manual ports	SN76489 backend	PSGlib_CV	BIOS-style sound tables
Sequenced music	BIOS format and Tiny Sound	`MUSIC`, simple/full players	External/player libraries	`MUSIC` backend	PSG streams; looping/status	NMI-serviced music sequences
Concurrent SFX	Coleco table areas/Tiny Sound	Channel depends on music mode	Library-dependent	Pending measurement	PSG SFX channels and frame service	Four music areas plus sound areas 5+ for SFX
Digital samples	DSOUND and adaptive VoxPCM, including WAV conversion and preview	No first-class support	Manual/custom	Pending	No first-class support found	No first-class support found
Speech hardware	SP0256-AL2 Lundy and EVE detection, state, queueing, timing, and debugger emulation	Manual I/O/custom	Manual I/O/custom	Not established	Manual I/O/custom	Manual I/O/custom
Authoring	Integrated table/SFX editors, two-channel Tiny Sound sequencer, MIDI capture, and WAV-to-VoxPCM workflow	Note-oriented BASIC source	External VGM/tools	Source/resource conversion	External PSG tools	External table/asset tools

Compression evidence

Compression must be scored as **`compressed payload + linked decompressor`**, with destination and cycles. Counting a host compressor without a ColecoVision decoder is invalid.

All sixteen integrated Amy codecs were also compiled as separate Balanced ROMs and run for 180 NTSC frames in GearColeco. Every ROM reproduced Warrior's 6,144 Pattern bytes and 6,144 Color bytes exactly in VRAM. Run **`node tools/test-integrated-codec-vram-roms.mjs`** to repeat this check.

Direct-to-VRAM speed ranking

This benchmark measures two complete 6,144-byte Amy decompression commands (Pattern and Color) with NMI disabled. GearColeco counts exact Z80 master-clock cycles between executable markers and then verifies all 12,288 VRAM bytes. The samples are the mechanically selected most-compressible, median, and least-compressible corpus pictures. Run **`node tools/benchmark-codec-vram-cycles.mjs`**.

Rank	Codec	Decoder bytes	Average cycles	Observed range	NTSC frames	PAL frames
1	MDK-RLE	46	445,125	397,660-480,429	7.45	6.24
2	MSC1	68	537,643	509,124-575,481	9.00	7.53
3	LZF	117	2,228,337	1,699,611-2,499,514	37.30	31.23
4	Nibble	115	2,500,600	1,109,498-4,605,253	41.86	35.04
5	BitBuster	166	2,534,796	2,200,282-2,848,517	42.43	35.52
6	ZX2	115	2,640,512	2,523,741-2,813,662	44.20	37.00
7	ZX1	127	2,649,192	2,542,424-2,824,747	44.35	37.12
8	ZX0	136	2,783,160	2,596,202-2,925,467	46.59	39.00
9	aPLib Compact	244	2,882,944	2,038,378-3,910,864	48.26	40.40
10	Pletter	212	3,011,945	2,593,207-3,420,091	50.42	42.21
11	ZX7	136	3,045,916	2,646,484-3,421,759	50.99	42.68
12	DAN3	205	3,260,613	2,819,166-3,514,464	54.58	45.69
13	MegaLZ	162	3,545,015	3,003,576-4,081,136	59.35	49.68
14	DAN2	212	3,683,817	2,806,751-4,725,201	61.67	51.62
15	DAN1	205	3,809,350	2,827,472-5,013,092	63.77	53.38
16	Exomizer 2	226	5,443,620	3,379,560-8,078,404	91.13	76.28

Frame values are time equivalents (**`59,736`** cycles NTSC; **`71,364`** PAL), not VBlank waits. Speed and ROM-size rankings are separate: choose using compressed payload plus decoder size, then check whether the measured loading time suits the game. Nibble and aPLib vary substantially by input, which is why Amy Studio should present per-asset estimates rather than one universal winner.

Solution	Confirmed formats	Direct VRAM status	Integrated selection
Amy Studio	Exomizer 2, ZX0, ZX1, ZX2, aPLib, MegaLZ, MSC1, ZX7, Pletter, DAN1/2/3, LZF, BitBuster, MDK-RLE, Nibble	ColecoVision paths, including workspace-based formats	Browser comparison/import and asset metadata
CVBasic	Pletter	`DEFINE CHAR/COLOR/SPRITE/VRAM PLETTER`	Explicit source keyword
z88dk	ZX0/1/2/7 and aPLib families, multiple speed/size decoders	Stock decoders target RAM; ZX0 *and ZX7* Coleco VRAM adaptations verified here	Manual headers/linking and host tools
ugBASIC	MSC1 and RLE types in compiler source	MSC1 image fallback verified; RLE is not implemented for Coleco	Resource compiler can choose compression when it wins
SGlib_CV / SMSlib routine	ZX7 and aPLib	ZX7 API; aPLib direct-to-VRAM exact with frame IRQ disabled	Manual host compression and C asset inclusion
PVColLib	RLE, Pletter, DAN1/2/3	RLE direct-to-VRAM verified exactly here; other paths remain separate candidates	`gfx2col` conversion and C asset inclusion
NewColeco	GETPUT 1.1 MDK-RLE	Direct-to-VRAM verified exactly on Warrior	External CVPaint/asset tools and C data inclusion

z88dk's public ZX0 integration request dates from February 2021, consistent with ZX0/ZX7 being available there by spring 2021. Its bundled compressor identifies itself as ZX0 v1.5 and emits the official classic v1 format. Amy Studio's current `zx0` codec emits the later official v2 format by default. The streams are intentionally not interchangeable. z88dk's stock routines target RAM; this benchmark adds only ColecoVision TMS9918 direct-to-VRAM adaptations and validates their output independently.

Codec naming and integration policy

Candidate	Honest Amy name	Current evidence	Integration condition
ZX0	`ZX0` / codec `zx0`	Existing v2 browser encoder and Coleco VRAM decoder	Keep as the default
ZX1	`ZX1` / codec `zx1`	Byte-identical browser encoder; exact GearColeco VRAM and cycle proof	Integrated and measured
ZX2	`ZX2` / codec `zx2`	Exact corpus round-trip, five-profile VRAM proof, and cycle proof	Integrated and measured
aPLib	`aPLib` / codec `aplib`	Bidirectional appack parity and exact Amy/GearColeco VRAM ROM	Integrated; keep NMI-safe upload and attribution explicit
MegaLZ	`MegaLZ` / codec `megalz`	Independent DEC40-compatible encoder; exact corpus round-trip and GearColeco VRAM/cycle proof	Integrated and measured
Exomizer 2	codec `exomizer`	Independent browser encoder, exact 42-picture round trip, five-profile VRAM proof, and cycle proof	Integrated; 226-byte decoder; compiler-assigned 256-byte-aligned `AMY_EXOMIZER_TABLE` reserves 156 bytes through address +$009B
MSC1	`MSC1` / codec `msc1`	Exact 42-picture round trip and GearColeco VRAM proof	Integrated; 68-byte direct-to-VRAM decoder

A host compressor alone is insufficient. An Amy codec requires round-trip tests, exact GearColeco VRAM output, decoder cost, explicit destination semantics, malformed-stream failure tests, documentation, and attribution. RAM-only APIs and locally written VRAM ports must remain labelled.

Current measured status: Amy Studio integrates the broadest codec selection examined in this ColecoVision workflow; devkitSMS has verified ZX7 and aPLib paths, PVColLib has a verified RLE path, and CVBasic has a concise Pletter path. On Warrior and Cake, official ZX1, ZX2, and aPLib do not beat Amy's ZX0 first-use ROM size. aPLib nevertheless cuts the devkitSMS Warrior ROM from 13,680 raw bytes to 4,713 occupied bytes. ZX1 is now an integrated Amy codec with a 127-byte direct-to-VRAM decoder and exact runtime and cycle proofs. ugBASIC's MSC1 is a valid comparison candidate, not ten separate codecs. Amy Studio now exposes the same stream family independently; it does not improve the Warrior resource over ZX0.

What MSC1 actually is

MSC1 is ugBASIC's block-repetition format, not a general ZX0-style LZ codec. Tokens encode 1-127 literals or repeat one four-byte sequence up to 32 times using a roughly 1 KB window; zero ends the stream. It favors data containing repeated four-byte groups.

ugBASIC's current Coleco image path includes a direct-to-VRAM MSC1 decoder. Amy Studio's independent 68-byte implementation likewise selects the TMS9918A write address once and streams literal bytes or repeated four-byte sequences without reading VRAM. Both the JavaScript codec and generated ROM pass exact round-trip and GearColeco VRAM checks.

Across 42 Graphics II images, MSC1 totals 259,023 bytes, averages 50.19%, and has a 45.63% median. It ranks 16th and records no top-five first-use finish. Warrior needs 3,716 payload bytes, or 3,784 with the decoder. MSC1 favors decode speed over density; ZX0 remains Amy Studio's better default.

aPLib integration status

aPLib has three useful pieces already available locally: the official host compressor, Amy's JavaScript beam-search encoder, and direct-to-VRAM Z80 implementations in z88dk/devkitSMS. The devkitSMS routine uses the Coleco-compatible **`$BF/$BE`** VDP ports, but is explicitly unsafe while the display is active and lacks the Coleco BIOS protection used by **`SG_decompressZX7toVRAM`**.

The JavaScript encoder now passes bidirectional differential checks against official aPLib on real TMS9918 tables and structured, incompressible, repetitive, short, and boundary-sized data. Its exact-cost planner produces 3,054 bytes for Cake and 2,973 for Warrior, versus official raw payloads of 3,088 and 2,975. Official **`appack`** still wins some individual resources, so neither encoder universally dominates.

Amy now ships the browser encoder and a 244-byte compact public-domain SMSlib-derived ColecoVision VRAM decoder. **`decompress aplib Source to vram.pattern`** accepts a raw **`.aplib`** stream and the existing VRAM upload wrapper provides the same NMI-safe contract as other Amy LZ codecs. A compiled Amy ROM reconstructed Warrior's 6,144-byte pattern and color tables exactly in GearColeco under all five optimization profiles. The Balanced test ROM occupied 3,517 bytes with 2,973 compressed data bytes. The official aPLib executable is used only for differential testing and is not redistributed.

The first reproducible size results and the rules for the cross-tool runtime comparison are in **`competition/benchmarks/compression/README.md`**. On two real 12,288-byte TMS9918 pictures, ZX0 has the smallest Amy first-use ROM total after its current 136-byte decompressor estimate is included. The separate GearColeco cycle benchmark supplies the speed evidence; size and speed remain independent selection criteria.

Memory, ROM size, and banking

Area	Amy Studio	CVBasic	z88dk	ugBASIC	devkitSMS	PVColLib
Stock 1 KB RAM focus	Core design, RAM estimates, overlays	Yes, global/static model	Configurable CRT/C runtime	Backend manages runtime/resources	SDCC/static library model	SDCC/static library model
Dead helper elimination	Capability-driven generation	Compiler-generated runtime	Linker sections/libraries	Deploy-on-use modules and target optimizer	Linker library extraction	Linker library extraction
Optimizer	Five profiles plus runtime corpus	Z80 optimizer and peepholes	sccz80/zsdcc optimizers	Coleco-specific optimizer source	SDCC optimizer/peepholes	SDCC size optimization
Beyond 32 KB	MegaCart images, declared bank outputs, fixed trampolines, bank-qualified symbols/maps, breakpoints, trace, rewind, and profiling verified from 64 KB through 1 MB	MegaCart up to 1 MB	Coleco banking/toolchain	Target support not yet proven	MegaCart and banked functions documented	MegaCart tools and example verified
Debug-aware RAM names	Yes, including overlay aliases	Assembly labels	Map/debug symbols	Generated symbols	Map symbols	Map symbols

Banking is established in CVBasic, z88dk, devkitSMS, and PVColLib. Amy Studio now builds and runtime-validates 64-1,024 KB MegaCart projects with separate fixed and switchable outputs, bank-qualified source maps, and bank-aware debugging. Ordinary `include amy` files remain one textual compilation unit; relocatable monolithic bank sections are a separate future ABI. Stock unbanked cartridges remain the size-ranking baseline.

Stock baseline versus expanded hardware

The size and runtime ranking above targets an original ColecoVision: TMS9918A video, SN76489 sound, 1 KB system RAM, Coleco BIOS, and an unbanked cartridge whenever possible. Optional hardware is credited separately because it changes the machine available to the programmer:

Extension	Verified or documented advantage	Scope in this comparison
MegaCart / bank switching	CVBasic, z88dk, devkitSMS, and PVColLib can exceed the normal cartridge space	Valid capability; excluded from stock-ROM size ranking
F18A	PVColLib provides APIs and examples; Amy's GearColeco 1.7.0 debugger can emulate F18A video	Debug validation is available in Amy, but first-class Amy language APIs remain future work
SGM / AY-3-8910-compatible sound	CVBasic and PVColLib provide explicit SGM paths; PVColLib also detects SGM RAM and ADAM	Valid expansion/clone capability; out of scope for the stock sound ranking
SP0256 voice modules	Amy provides Lundy/EVE commands and debugger-selectable module emulation	Optional external hardware; excluded from the stock sound ranking
Coleco ADAM	Amy builds native EOS and OS7+EOS hybrid programs on DSK/DDP, with target-aware graphics, input, sound, compression, files, ADAMnet, serial, and modem services	Broad DSK/DDP runtime coverage; applications beyond the resident 6 KiB window and optional-EOS cartridge services remain architectural work
Extra RAM	Available through SGM, ADAM, and compatible clones depending on the tool/runtime	Report separately; never count it as stock 1 KB RAM

This separation lets every solution show its extended-hardware strengths without weakening Amy Studio's deliberate original-hardware target.

Development experience

Area	Amy Studio	CVBasic	z88dk	ugBASIC	devkitSMS	PVColLib
Browser IDE	Integrated	No official integrated IDE	No	IDE/web options advertised	No	No
Source breakpoints	Yes	External emulator	External debugger	IDE-dependent/unverified	External emulator	External emulator
ASM stepping/rewind	Integrated GearColeco 1.7.0 workflow	External	External	External	External	External
RAM/VRAM/symbol inspection	Integrated	External	External	External	External	External/map symbols
Cycle profiling	Integrated	External/manual	External/manual	Unverified	External/manual	External/manual
Project graphics editors	Configurable `editors.json`; bitmap, tile, sprite, frame, tilemap, and composite previews	Separate tools	Separate tools	Resource conversion/IDE	Separate tools	`gfx2col`/separate tools
External ROM loading	Drag-and-drop `.rom` and `.col` into Debug	External emulator	External emulator	External emulator	External emulator	External emulator
Optional hardware setup	Voice-module selection plus browser-local Coleco/ADAM firmware	External emulator	External emulator	External emulator	External emulator	External emulator
Automated ROM tests	Corpus, checkpoints, runtime harness	Not integrated	Buildable manually	Not established	Not established	Not established

Amy Studio integrates this workflow; command-line-only Amy scripts remain labelled separately from features available directly in the IDE.

Amy Studio emphasizes ColecoVision integration

Amy Studio connects source editing, assets, compression, assembly optimization, ROM execution, external-ROM loading, breakpoints, rewind, memory/VRAM inspection, hardware configuration, profiling, and automated tests in one ColecoVision workflow. Its graphics editor follows TMS9918 sprite priority and scanline rules while previewing tile/sprite composites. Records, overlays, typed state machines, BCD, fixed-point and checked wide integers, collision helpers, and BIOS-aware input are additional language-level strengths.

CVBasic emphasizes portability and established BASIC simplicity

CVBasic combines a compact QBasic-like surface with a mature ColecoVision backend, complete game examples, spinner and sprite-flicker support, music commands, optional ROM banking, and several related targets. It does not offer Amy's records, overlays, wide numeric model, integrated debugger, or asset/debug pipeline.

Official source: nanochess/CVBasic

z88dk emphasizes general C and toolchain breadth

z88dk provides C data structures and pointers, mature compilers, assemblers, linkers, libraries, compression utilities, and many Z80 targets. Its flexibility exposes more low-level choices; generic facilities require target-specific proof before comparison with Amy's ColecoVision APIs.

Official source: z88dk/z88dk

ugBASIC has the most ambitious portable high-level vocabulary

ugBASIC documents nonblocking animation, movement, paths, image conversion, sprites, sound, and many modern BASIC constructs. This makes it especially relevant when considering future Amy animation ergonomics. However, its multi-target manual describes capabilities that can differ by backend. Generated ColecoVision Z80 and runtime behavior must be measured before declaring those features equivalent.

Official sources: ugBASIC site and spotlessmind1975/ugbasic

devkitSMS offers a compact C game-library workflow

devkitSMS combines SDCC with small game-oriented libraries and asset tools. **`SGlib_CV`** and **`PSGlib_CV`** make it relevant to ColecoVision even though much of the surrounding documentation is SMS-oriented. Only APIs present in the CV headers and libraries should enter the final score.

The local **`SGlib_CV`** source confirms that its ColecoVision NMI scans direction/fire mode and numpad/right-trigger mode, preserves current and previous states, and implements **`SG_getKeysStatus`**, **`SG_getKeysPressed`**, **`SG_getKeysHeld`**, **`SG_getKeysReleased`**, and **`SG_readCVNumPad`**. These input capabilities are source-verified for ColecoVision, not inferred from SMSlib.

Official source: sverx/devkitSMS

PVColLib offers broad ColecoVision-focused C APIs

PVColLib is a ColecoVision-focused SDCC library and development kit. Its official repository includes the compiler toolchain, VDP and controller APIs, sprites, sound/music support, compression routines, MegaCart, SGM, Phoenix, and F18A facilities. The three stock fixtures here use its official build flags and linked library. Local headers and examples verify both controller ports and keypads, both spinners, 32 sprites, NMI sprite swapping, hitbox collisions, BIOS-style sound tables, sequenced music with concurrent SFX areas, direct VRAM operations, MegaCart, SGM, Phoenix, and F18A support. Its RLE bitmap path was validated byte-for-byte in VRAM; the available Pletter and DAN paths are not credited with a benchmark size until a compatible stream passes the same exact runtime test.

Official source: alekmaul/pvcollib

NewColeco provides a historical baseline

The recovered SDCC branch uses Amy's original **`CRTCV`**, **`CVLIB`**, and GETPUT 1.1 libraries. Their prebuilt ASxxxx objects require the historical stack-parameter ABI, **`.o`** object naming, and first-object linker output convention. The comparison pins SDCC 2.9.0 #5416 (March 2009). Modern SDCC 3.8/4.5 builds can link and boot while silently miscalling stack-ABI routines, so only behavior-validated NewColeco results count here. Warrior is decompressed directly to VRAM with GETPUT's MDK-RLE routine and matches all 12,288 target table bytes and all 49,152 pixels.

This is not an unrelated additional competitor: it is the documented historical ancestor of Amy Studio. Its 795-byte Hello, 3,643-byte DAN2 bitmap, and 932-byte controller monitor show that the old BIOS-aware and link-only-what-is-used philosophy was already effective. Amy Studio adds the Amy language, integrated assets, diagnostics, and debugging; this describes an evolution of workflow, not a judgment on the earlier kit.

PkK's devkit is compact but deliberately low level

Philipp Klaus Krause's devkit uses **`libcv`** / **`libcvu`**, links only used SDCC modules, and completes all six fixtures. Its six-sample total is 10,760 bytes, third among the ranked toolchains. The deterministic state update averages 7,707 cycles with a 9,346-cycle worst frame. Its native RLE+Huffman bitmap path reproduces all 12,288 Warrior bytes exactly, but the 2,934-byte payload also needs a 514-byte tree/configuration, about 411 linked decoder bytes, and 19 bytes of runtime state. Across 42 pictures its median payload is 4,098 bytes and it usually ranks 14th or 15th among 17 measured methods. This makes PkK's devkit useful evidence for compact C linking and shared-codebook research, not a general replacement for Amy's current bitmap codecs.

Next measurement suite

The comparison becomes stronger as equivalent game behaviors are implemented eight times. Current status and order:

1. **Controller snapshot: complete.** Seven toolchains pass injected neutral, keypad, direction,

fire, and release states; ugBASIC's VDP R7 update remains the documented exception.

2. **Sprite/metasprite stress: complete.** Eight toolchains build and boot the shared fixture.

Amy's native metasprite path is runtime-equivalent, uses zero permanent RAM, preserves explicit layer priority, and is smaller than the manual renderer in all five profiles.

3. **Tile animation: complete.** Eight ROMs animate shared patterns and 3x2 Name Table frames;

GearColeco verifies every pattern, color, tile position, animation phase, and final state.

4. **State update: complete.** Eight ROMs run the same actor array, collision checks, timers, and

state dispatch oracle.

5. **Sound and voice: functional and still evolving.** BIOS commands and Tiny Sound sequences can

be inspected, auditioned, edited, imported, and written back. VoxPCM WAV conversion and preview, plus SP0256 Lundy/EVE commands and emulation, are integrated; UX and hardware-parity QA continue.

6. **Compression: payload and cycle suites complete.** Fifteen codecs have exact VRAM proof;

representative streams now include linked decoder bytes and GearColeco cycle measurements.

7. **Visible Hello: complete.** Keep it separate from the minimal runtime fixture so font/text

costs remain explicit.

All six Amy listings are available in Amy Studio under **Benchmarks**. The evidence archive groups the matching source and loadable ROM for every tool. Measurements use occupied bytes rather than padding, and a smaller result counts only when the shared runtime oracle passes.

Ranked Amy gap plan

Rank	Open work	Status	Value	Effort	Risk
1	Extend the completed declared-output MegaCart model with relocatable monolithic bank sections and parameter/return ABI	64 KB-1 MB packaging, fixed trampolines, data access, bank-qualified maps, breakpoints, trace, rewind, and profiling are verified	Medium for ports that cannot separate bank files	Large	High
2	Extend native EOS programs beyond the resident `$C800-$DFFF` window	Native/hybrid DSK/DDP, EOS APIs, files, ADAMnet, serial, and modem paths are runtime verified	High for larger ADAM software	Large	High
3	Add optional EOS services to otherwise stock OS7 cartridges	Requires safe detection, low-RAM trampoline, MIOC restoration, and stock-console fallback	High for saves and shared releases	Large	High
4	Close sound-editor and hardware-fidelity workflow gaps	Active	High	Medium	Medium
5	Add a small explicit runtime animation service	Implementation study; editors already support animation data	High	Large	Medium-high

MegaCart's image, mapper, linker, and debugger gates pass at 64, 128, 256, 512, and 1,024 KB. Repeated logical addresses retain physical-bank identity through fixed-bank calls, source maps, breakpoints, trace, rewind, profiling, and capacity diagnostics. The remaining optional extension is a relocatable-object ABI for monolithic `bank n` sections and richer parameter/return calls; declared independent bank outputs do not depend on that extension.

The codec study starts from the verified `libcv` RLE+Huffman path in PkK's devkit without copying its format into Amy. It will test a compact canonical or fixed shared codebook plus project-level escape analysis across all 42 pictures. Adoption requires exact JavaScript round trips, direct-to-VRAM Z80 proof, measured decoder bytes, CPU RAM, GearColeco cycles, and a first-use ROM win over Amy's existing codecs.

Completed gaps and next concrete work

Completed and runtime-guarded:

- exact nine-tool metasprite, state-update, and tile-animation runtime oracles, plus native

metasprites, protected-priority flicker, and editable graphics;

- complete 2D arrays and record/overlay fields with bounds diagnostics and five-profile proof;
- sixteen direct-to-VRAM codecs with round trips, decoder cost, cycles, RAM, and 42-picture ranks;
- corrected indexed `put frame`, sound inspection/editing, playback, undo, import, and Web MIDI;
- adaptive VoxPCM conversion, preview, reusable-word playback, Space Taxi/Solar System demos, and

SP0256-AL2 Lundy/EVE commands, timing, queues, and debugger emulation;

- GearColeco 1.7.0 with F18A/ADAM emulation and browser-local OS7/EOS/WP firmware; native EOS and

OS7+EOS hybrid DSK/DDP targets now have runtime-tested graphics, input, sound, compression, filesystem, ADAMnet, printer, serial, and modem paths;

- `.rom`/`.col` drag-and-drop and universal graphics editors with TMS9918 priority and scanline

preview, tilemaps, frames, and tile/sprite composites;

- 155 five-profile display tests; adaptive `u8` saves 212 corpus bytes without ROM growth, while

explicit Amy digits match hand-written Z80 at 498 bytes and 1,905 cycles.