Structris — CVBasic, TI-99/4A & ColecoVision

"Inverted Tetris": the machine builds the structure and throws pieces at *you*. Climb, duck, and
dodge along the skyline of a growing stack of blocks — don't get buried. Built in **CVBasic**,
dual-target from one source: TI-99/4A (native TMS9900 bank-switched cartridge ROM, `--ti994a`)
and ColecoVision (native Z80 ROM, CVBasic's default target) — not an XB256/compiler game.
Full spec in [`DESIGN.md`](DESIGN.md).

Original concept, piece catalog, and neighbor-height targeting AI: **Martin Haye**, 2010 (Apple II
/ Applesoft BASIC), introduced at KansasFest —
github.com/martinhaye/structris.

Play

- You stand on top of the block stack. A continuous **stream** of pieces (1-3 columns wide, up to
  6 in the shaft at once with ~1.3 rows of air between them) falls smoothly from above the screen
  top, each aimed at wherever you're standing when it spawns, permanently adding to the stack
  where it lands.
- You move **smoothly, 2 pixels per frame** (like the pieces fall), and you're a squeezed 4×2-px
  bar rather than a full-cell character: slide along the skyline, climb into open shaft, duck into
  a valley — falling pieces block movement just like settled blocks, pixel-exact, and being only
  2 px tall you can **squeeze through the ~11-px gaps** in the stream.
- A piece descending onto you **forces you down**, pixel by pixel; when there's no room left below
  (the stack, the floor, or another piece), you're **smashed**: game over. You can ride down
  inside a gap between pieces and slip out sideways — but a piece landing on you closes the gap to
  nothing, so get out from under it — a piece that lands *on* you buries you even at the fastest
  fall speeds. When you're buried, your bar stays on screen **blinking** at the spot where it
  happened and the HUD/backdrop turns dark red; fire returns to the title. Surviving all 10
  levels earns a dark-green victory banner instead.
- Rows needed per level: 7 at level 1, +2 each level (25 at level 10).
- Scoring: 1 point per landed piece; clearing lines pays 10 (single), 50 (double), or 100
  (triple — the maximum, since a clear is capped by one bar's height) — the bonus

rewards
  simultaneous rows. The score, level, and clear progress live in a **left sidebar** (score
  top-left, `LEVEL` and `CLEAR` blocks under it); the score spans the whole game (all levels).
- Every column built up past a shared height threshold "falls away" (the completed rows compact
  out) and counts toward the level. Clear enough rows and the level advances: the shaft narrows
  (15 columns at level 1 down to 6 at level 10) and the row goal climbs,
  but the **fall speed stays constant** (a faster fall made high levels unplayable — the challenge
  comes from the row requirement and the narrowing shaft, not from speed). The
  checkered **mountain** the game stands on (narrow at the top under the walls, flaring one column
  per side every row down to a wide base at the floor) rises half a character each level — so it
  grows taller and wider as the shaft shrinks, shortening the fall from 16 rows at level 1 to 11 at
  level 10. The mountain is revealed column by column as
  the walls slide open between levels.
  Survive all 10 levels to win.
- **Between levels:** a **flush** plays first — the shaft interior clears under a descending tone —
  then the walls slide to the new width. The two targets clear it differently for speed reasons:
  ColecoVision does the full drain (any pieces still falling first snap onto the grid in their
  colors, then the whole stack slides down and vanishes just above the mountain — cheap on the Z80),
  while the **TI-99** does a lighter **wipe** (blank the interior one row at a time from the top
  downward), because the full row-shift drain does far too many VDP round-trips per frame to run
  smoothly on the slower TMS9900. Same descending tone on both; the mountain foundation is never
  touched either way.

Controls (joystick 1)

Action	Input
Slide left / right (2 px/frame, frame-paced so TI matches Coleco speed under load)	Joystick left / right
Climb / duck (same frame-paced step)	Joystick up / down
Start (title) / back to title (game over, win)	Button
Setup: pick starting level	Type `8` `3` `8` on the title, then `1`–`9` (or `0` for level 10) — the game starts immediately. Lasts one game only; the title always resets to level 1.

How it's built

- CVBasic **default video mode** (no `MODE` call — `MODE 2` compiles but renders broken on both
  machines; see `DESIGN.md` header) with 8 solid tiles at chars 128–135: seven piece colors plus
  the white border, colored per-character via `DEFINE COLOR` like every other CVBasic game in this
  repo — plus **2× sprite magnification** (`VDP(1) = $E3`, as in Astiroids).
- **Every falling piece is ONE magnified sprite descending 1 pixel per frame** (every shape in
  the catalog fits a 3×3-cell box, so a 32×32 magnified sprite holds any piece; its 16×16 def is
  composed in VRAM at spawn). On landing the sprite is hidden and the piece is converted to
  background characters in place. The player is a sprite too: a 4×2-px white bar at a free pixel
  position, moving 2 px/frame with pixel-exact collision (the squeeze is what lets it slip
  through the stream gaps). The shaft ceiling sits at the screen top so pieces enter smoothly
  from above the display.
- The **piece catalog and targeting AI are ported directly** from the original's
  `ON HL*4+HR+1 GOTO …` table: pieces are picked by comparing a target column's height to its two
  neighbors, so every piece **fits the terrain perfectly** — it falls as one rigid
  tetromino-like shape (all columns sharing a single pixel fall position) and slots flush onto
  the skyline in one landing. See `DESIGN.md` §5–6 for the full table and what got reinterpreted
  (the Apple II renderer's continuous hardware scroll has no TI-99 equivalent).
- **Height forecasting** makes the stream land correctly: targeting/shape selection read a
  forecast array (settled heights + every in-flight piece's booking), so a later piece aims at and
  fits the surface as it *will* be — the original books heights at emission time the same way.
- The smash rule is the original's collision verbatim: an occupied player cell forces the player
  down one row; a blocked cell below means death.
- **Same real-world speed on both machines.** The TMS9900 backend is slower than the Z80's, so the
  hot paths precompute everything at spawn/landing (pure compares/adds per frame) and the fall is
  paced by the elapsed `FRAME` delta — a slipped frame becomes a 1-px catch-up step instead of a
  slowdown. Measured at 60Hz in Classic99 (see `DESIGN.md` header). No `#IF` platform branching
  needed anywhere in `STRUCTRS.bas`.

Music

Every level has its own tune (`tune1`..`tune10`) — ten original loops that get faster and
darker as you climb: level 1 is a 16-bar A-minor folk dance, levels 2–10 are shorter loops in
different keys/modes (D minor, E phrygian, G major, C dorian, A harmonic minor, F# minor, B
phrygian, E harmonic minor, A minor) at rising tempos. They play via CVBasic's interrupt music
player — melody + bass on channels 0/1, effects on channel 2, noise reserved for explosions and
fireworks. A game starts with a **3-2-1 countdown** (rising beeps, one per second, centered over
the shaft); when the "1" clears, the level's tune and the piece stream begin together.

Build

- **TI-99/4A:** `bash build-ti.sh` → `src/STRUCTRS_8.bin` (load in Classic99 or js99er).
- **ColecoVision:** `bash build-coleco.sh` → `src/structrs.rom` (load in CoolCV or blueMSX).

> **Requires the forked `cvbasic`** (`unhuman/CVBasic`). The source uses `#if TI994A … #else …
> #endif` to pick the level-up clear per target (TI wipe / ColecoVision bake+drain); the fork adds
> those directives and auto-defines `TI994A=1` on `--ti994a`. Stock nanochess CVBasic has no
> preprocessor and will not build it. No `-D` flags are needed. See DESIGN header + §10.

> **TI-99 program budget is a hard 24,336 bytes** (single-bank; not the 32 KB cart size). Past it,
> `linkticart` silently truncates the tail → visual corruption. Report free bytes after every build
> and reclaim from code (not the ~2.8 KB of music) first. See DESIGN §10.

Status

TI-99 build verified playing in Classic99 (scripted emulator runs: title → fire → pieces
streaming as single magnified sprites falling pixel-by-pixel with ~11-px gaps, smooth entry from
above the screen top, landings converting flush to per-piece colored tiles, the player bar
sliding/climbing smoothly in pixels and surviving mid-stream between pieces, smash rule kills a
player caught under a landing piece → OOPS with blinking player → fire returns to

the title;
838 → level 4 start verified, and the starting level resets to 1 after a game (fire → title →
fire starts level 1); the flared **mountain** stage renders cleanly (verified at levels 1/7/10 and
across forced level-up transitions) with the walls meeting the checker (no black gap) and reveals
column-by-column across chained level-ups; the
player bar starts centered and the OOPS!/LEVEL UP! messages sit centered under the tower; the
green win banner and its color restore verified via a probe build; 60Hz timing confirmed by
blink-period and fall-rate measurement).
ColecoVision build compiles clean and shares all the same game logic; needs a CoolCV pass.

Two toolchain landmines were found (and are documented in `DESIGN.md`'s header): CVBasic's
`MODE 2` renders broken on both machines (use the default mode + `DEFINE COLOR`), and the
CVBasic 0.9.2 **TI-99 backend miscompiles `<comparison> AND <comparison>` conditions**
(stale-register AND → the original instant-OOPS bug) — every condition in this game is written
as nested single-comparison IFs.