CZUBATRONIC Retro Chess System

Product Overview

CZUBATRONIC is a standalone retro-styled electronic chess system for playing, studying, analyzing, replaying, and saving chess games. It combines a graphical chessboard, UCI engine support, configurable play modes, position setup, PGN storage, replay navigation, engine analysis, selectable piece sets, move indicators, clocks, and engine telemetry.

The interface is designed as a 1980s professional electronic chess computer: a wood-textured board, dedicated status panels, move notation, engine output, and hardware-style controls.

Starting the Application

There are two different ways to run CZUBATRONIC. The Python command below is for developers running the source project. It is not required for customers who receive the packaged Windows executable.

Developer / Source Run

From the project directory with the virtual environment active:

```
python -m retro_chess_app.retro_app
```

The application opens with a system-check splash screen. Click the splash screen, press a key, or wait for it to dismiss automatically.

Customer / Installed Application

A PyInstaller one-folder build (`CZUBATRONIC.exe` plus `_internal\`) is installed by the official Inno Setup package. Customers do not need to install Python, create a virtual environment, install packages, configure engines, or run a Python command. Start the product from the Start menu or desktop shortcut.

Windows may show its normal download or SmartScreen warning for an unsigned installer. The installer also provides the current end-user manual, release notes, and the End User License Agreement.

The build was a one-file executable until the first 0.9.1 builds. A one-file EXE runs as a bootloader parent plus the real application; when Inno Setup's "closing applications" step (the Windows Restart Manager) closed a running copy, the application exited but the bootloader survived without a window and kept `CZUBATRONIC.exe` locked, so every upgrade failed with `DeleteFile failed; code 5`. The one-folder build is a single process, starts faster and unpacks nothing into `%TEMP%`. `tools\rm_shutdown_check.ps1` reproduces the Restart Manager shutdown against a build and fails if any process survives; the installer additionally detects a still-running copy (visible or not) and offers to end it before installing.

Portable Edition

`build_release.ps1` also zips the same one-folder build with the documentation and the End User License Agreement as `CZUBATRONIC_2400_<version>_Portable.zip` (archived under a build-time/commit-tagged name like the installer). It runs from any writable folder without installation or administrator rights. The marker file `CZUBATRONIC_PORTABLE.txt` beside `CZUBATRONIC.exe` switches `user_data_directory()` to `<program folder>\config`, so the remembered standard engine and the diagnostic log stay with the program; without the marker the installed-edition location `%LOCALAPPDATA%\CZUBATRONIC` is used.

Diagnostic Log

`diagnostic_log()` appends timestamped lines to `czubatronic.log` in `user_data_directory()`: application start with version, engine close, window destroy, main-loop end, and every exception raised inside a Tk callback (which a windowed EXE would otherwise swallow silently). The file is truncated when it exceeds 1 MB.

Engine Requirement for Distribution

The customer build bundles Pioneer, CAPA96, Tacticon, and Zukertort. CAPA96 must retain its `ENGOPEN.TXT` companion file beside `CAPA96.exe` because its opening data is loaded from the engine working directory. Tacticon is bundled as the self-contained `_internal\engine\tacticon\Tacticon.exe`, Zukertort (a historical personality engine from playchessgate.com, with `Skill Level` and `Style Weight` UCI options) as the self-contained `_internal\engine\zukertort\Zukertort.exe`; it writes a small `zukertort_uci.log` beside itself. The `ENGINE` selector also lets a customer browse for another compatible local UCI engine. The engine itself must be available for engine moves, hints, and analysis; the Python runtime is not required.

Opening books are likewise engine-specific. If an engine supports an external book option, the customer can select the book through `BOOK`; otherwise the engine may use its own built-in book.

Playing Chess

Human vs Engine

Play against the active UCI chess engine. The engine moves automatically whenever it is the engine's turn.

The player can choose to play as White or Black from the `NEW` game dialog. When playing Black, the engine correctly makes the opening White move before the player's turn.

Human vs Human

Use the same board for two human players. Engine moves are disabled.

Engine vs Engine

`MATCH` (the engine match key; the dialog is titled `ENGINE MATCH SETUP`) opens a compact setup dialog with a `GAMES` field for a series of up to 1000 games (colours alternating; each game appended as PGN to `<settings folder>\\matches\\<date>_<white>_vs_<black>.pgn`, the tally to `results.txt` in that folder, the running score in the `ENGINE MATCH` panel); it for independently selecting White and Black engines and one shared condition: fixed depth, time per move, or a tournament game clock (N moves in T minutes, then M moves in U minutes more, unused time carrying over). Once started, the setup closes, the normal board and notation display the game, and a fixed left panel shows separate output streams for both engines. The match can be stopped without replacing the engine selected for normal play.

Under a game clock the match runner keeps the time itself (`MatchClock` in `engine_match.py`): it measures the wall time around each engine call, sends `go wtime ... btime ... movestogo ...`, charges the elapsed time, adds the next period at the time control, and ends the game when a side oversteps (loss on time, or a draw when the opponent has no mating material). The chess clock panel shows both engines' times during the match, the PGN gets a `TimeControl` tag. After a search budget runs out the adapter sends `stop` and waits up to 60 seconds for the bestmove the engine owes: Capa 1.32 checks its clock only between iterations and acts on `stop` when the running iteration ends.

Legal Chess Rules

The chess model is provided by the DCS chess toolkit (`dcs_chess`, DCS Analytics' own library, which replaced `python-chess` in 0.9.37) and supports standard legal chess behavior, including:

- Legal move validation
- Check and checkmate

- Stalemate
- Draw by insufficient material
- Draw by the 75-move rule
- Draw by repetition
- Castling
- En passant
- Pawn promotion
- Side-to-move tracking

Piece Movement

Click a piece to select it, then click a legal destination square. Pawn promotion opens a promotion dialog with choices for:

- Queen
- Rook
- Bishop
- Knight

Illegal moves are rejected and do not alter the game.

New Game Configuration

Press `NEW` or the `N` key to open the new-game configuration dialog.

The dialog allows the player to choose:

- Side: White or Black
- A calibrated Pioneer level or an explicit non-Pioneer search control
- Game clock

Starting a new game always restores the standard starting position with White to move. It also exits position setup and replay mode cleanly.

Pioneer Engine Levels

The following strength profiles are calibrated specifically for Pioneer:

Level	Search Depth	Displayed Strength
Novice	2	1168 ELO
Beginner	3	1389 ELO
Club	5	1617 ELO
Expert	6	1859 ELO
Club Champion	7	1978 ELO
Master	10	2160 ELO
Grandmaster	12	2452 ELO

Each Pioneer level also has a calibrated thinking-time range to give the engine a deliberate playing feel. These ELO figures and profile names must not be applied to CAPA96 or another external UCI engine.

External Engine Controls

For CAPA96, Tacticon, Zukertort, and every other non-Pioneer UCI engine, `NEW GAME` and `ENGINE SETTINGS` use explicit controls rather than uncalibrated Pioneer labels: `FIXED DEPTH` (1-30), `TIME PER MOVE` (0.1-600 seconds, entered in seconds with tenths; converted to milliseconds only at the engine boundary), or `GAME CLOCK`. Game-clock mode allocates engine search time from the selected clock, including the increment in `3 MIN + 2 SEC`. The status panel shows `CONTROL` rather than a Pioneer level or ELO.

Thinking-Time Model (Human vs Engine only)

The engine's real search takes whatever its control allows - a fraction of a second at fixed depth 6, the full budget under `TIME PER MOVE` or `GAME CLOCK`. Game clocks are specs (`RetroChessApp.TIME_CONTROL_SPECS`, `CUSTOM...` via `open_custom_time_control`): seconds per move, or `moves` in `minutes` with an increment and a following period of `extra_moves` in `extra_minutes`; `_finish_clock_move` counts the moves to the control per side and credits the next period, `_engine_time_budget_ms` divides the remaining time by the real moves to go, and the PGN gets a `TimeControl` tag. UCI options are parsed with default, min, max and choices (`EngineAdapter._parse_uci_options`), set with `set_option` (setoption plus isready, refused during a search), re-applied after an engine restart, and stored per engine file under `uci_options` in `engine_config.json` (`_save_uci_options`, applied in `_activate_engine`). `MOVE NOW` during a search calls `EngineAdapter.stop_search()`: it sends `stop` only while a search is running (an idle engine may answer a stray `stop` with a second `bestmove`) and wakes the search loop; an engine that has not answered within `STOP_REPLY_SECONDS` (4 s), or one known to be deaf (`STOP_DEAF_ENGINES`; CAPA96 identifies as `Capa 9.6`), is killed, relaunched with its book option and asked for `go movetime 400` from the same position. Measured on CAPA96: after a `stop` mid-search no `bestmove`, no `readyok`, `quit` ignored. That restart is for UCI engines only. An emulated chess computer (`own_pace`) gets its `MOVE NOW` key through the plugin's `stop`: Franz Huber's plugin calls the interface's `stop_play()`, or its `start_play()` when there is no `stop_play` (the MM IV and MM V's `ENT`), and only while the machine is thinking for its own side. An interface file loads its base with `load_interface` in two forms, `return load_interface("mm4")` (the whole interface) and `interface = load_interface("polgar")` (extended by the file's own functions), and a clone set uses its parent's file; followed both ways, 118 of the 249 interface files of CB-Emu (October 2026) define `stop_play`, 130 fall back to `start_play`, and the one without either is the empty template. In THORSTEN'S CHOICE the press is `ENT` on the Glasgow, the MM V and the Milano (polgar.lua), `Go` on the Novag Expert, `Play` on the Analyst D++, `MOVE` on the Fidelity 2000, 2100, 2265 and 2325 (fdes2100d.lua), `PLAY` on the R30, and conc.lua's `stop_play` on the three Conchess. The console presses no move key of its own (0.9.27 doubled the Milano's and the Fidelity's for a day, after a survey that followed only the first include form). On MOVE NOW the search loop sends `stop` at once and again every `STOP_REPEAT_S` (6 s) until the bestmove arrives: the plugin presses the key when it can and drops a stop it cannot act on (measured 2026-10-08: a stop 0.5 s after our move was lost on the Fidelity 2100 and the Conchess and taken on the Milano; every computer of THORSTEN'S CHOICE answered its key within 5.2 s). 0.9.27 to 0.9.29 estimated the plugin's entry time instead (2 s per entered move plus a reset allowance) and held the stop back that long, which on a game of 25 moves loaded from a PGN came to 73 s (Franz Huber, 2026-10-10); no estimate is made any more. A `stop_search()` while `best_move_from_fen` is still setting the search up (`_search_starting`) is kept as `_interrupt_pending` and handed to the loop. **Digit displays from the layouts (0.9.31, stage one of the layout-driven reading):** `tools/display_layout_tables.py` maps every machine of MessChess.lst to its driver (`MessChess.exe -listsource`), fetches the driver and the layout from the MAME 0.220 sources (tag mame0220, drivers BSD-3-Clause, layouts CC0, cached under build/mame0220; a MessChess-only set takes its parent's layout from `-listclones`), reads `set_default_layout(layout_x)` per configuration with the calls between configurations followed, parses the layout's first view with params, repeats and groups expanded, and writes `retro_chess_app/data/display_layouts.json`: per machine the digit element kind (led7seg, led14seg, led16seg), the fields (digits grouped by proximity along the reading axis, rotation-aware: 90 degrees reads top-down), the separators (small lamps placed between two digits of a field: one lamp a point, two a colon; these are the digits' own point bits as the layout draws them) and the indicator lamps. `tools/validate_display_layouts.py <machines>` runs each machine headless, plays 1.e4, assembles the display per field with the table, and marks it

`validated` only when a field shows the plugin's move (E7E5, E7 E5, E7-E5) and, where there is a clock, a field counts. `MachineDisplay.digit_layout` (set by the adapter from `machine_display.digit_layout_of(machine)`, validated tables only) makes `_decoded` assemble per field (`_decoded_fields`: glyphs from the digit values' low seven bits, separators from the lamps, fields two spaces apart) and wins over the unread list. Validated on 2026-10-11 over all 156 tables (three passes, the check refined between them: any spelling of the move counts, the Boris's `E7■E5` included; a display that carries only a clock passes when the clock rises by small steps through the table, which a field in the wrong order cannot do): 135 validated, 127 by the move and 8 by the clock (a clock-only pass needs every field free of letters: the Prestige family's clock counted while its other field read `G5:0`, which may be a scrambled move, so it was withdrawn); 21 rejected: cc1, chessmstdm, feag, feag2100, feag2100a, feasbu, feasgla, fepriv, fpres, fpresa, fpresb, fpresbu, fpresc, fpresd, fprese, fpresgla (the fidel_elite family shows `G5:0` and never the move), fphantom, montreux, risc2500, risc2500a (three digit groups on one line that the table merges into one field), sc80 (a scrolling text). `--unvalidated` tries every table not yet tried, `--retry-rejected` the rejected ones again; the JSON is written after every machine. The width rule and `UNREADABLE_DIGIT_MACHINES` stay for machines without a validated table. `new_game(start_fen)` sends `ucinewgame` at once when the engine is idle (NEW, OPEN, SETUP pass the game's start position), so the plugin's reset runs while the user thinks; `_begin_game_if_new` then sends no second one. The search loop sends it only once the plugin has entered the moves on the board (`MACHINE_ENTRY_S` plus `MACHINE_ENTRY_PER_PLY_S` per ply, plus `MACHINE_RESET_S` after a `ucinewgame`, which makes the plugin reset the machine first) and then waits for the machine's answer at its own pace, never restarting it: measured on the MM V (8 October 2026), a `stop` inside the entry got no answer at all, one after it was answered in 3 to 5 s, and a restart boots the emulator and re-enters the whole game, after which the machine's own move was refused. The adapter also sends `isready` after `ucinewgame`, sends the game as `position startpos moves ...` (`position_command`, falling back to the FEN when the history does not reproduce it), drains stale output before every `go`, and decodes engine output as UTF-8 with replacement. Tacticon never searches beyond 7 plies (measured: `go depth 10`, `go movetime 6000` and a 10-minute clock all end at depth 7 after about 1.7 s), so its effective control range is depth 1 to 7. In Human vs Engine mode the move is then shown no earlier than a human-feeling thinking time computed by `RetroChessApp._human_think_time_ms`; engine matches, hints and analysis are never delayed, and `MOVE NOW` plays a pending move at once.

Step	Rule (constants in `HUMAN_THINK`)
Base window	Pioneer: the level's calibrated `think_min_ms`-`think_max_ms`; other engines: 2.5-6.0 s, uniformly random
Forced move (one legal move)	x 0.15
Check with at most three replies	x 0.45
Recapture on the square just captured on	x 0.35
Book move or within the first 8 moves	x 0.35
Endgame (12 pieces or fewer)	x 0.8
Complex middlegame (35+ legal moves and 6+ captures)	x 1.4
Long think	8 % of moves, x 2.2
Limits	0.4 s minimum, 15 s maximum
Clock	never more than 60 % of the engine's time budget for the move - its movetime, or the clock's per-move budget when a clock runs with a fixed-depth control - so the engine cannot overspend its clock; the search time already used counts against the target

Time Modes

The available clock modes are:

- No Clock
- 30 seconds per move
- 15 seconds per move
- 3 minutes plus 2 seconds increment
- 5 minutes per game
- 15 minutes per game
- 30 minutes per game
- 40 moves / 120 minutes

The clock state is displayed in the computer-status panel when a timed mode is active.

The 15-second and 30-second per-move modes are advisory chess-computer test controls. Reaching zero does not end the game; the clock resets after the move. Whole-game controls retain normal time-forfeit behavior.

When a clock mode is active, the engine receives a UCI time budget and searches for the available time. Timed searches do not use a fixed depth. The fixed level depth profiles are used only by untimed (`No Clock`) games.

Position Setup

Press SETUP to edit the board position directly.

Setup mode supports:

- Dragging pieces on the board
- Removing pieces with the right mouse button
- Selecting a piece from the setup piece menu
- Placing pieces on empty squares
- Choosing White or Black to move
- Choosing whether the human plays White or Black
- Selecting calibrated Pioneer strength or explicit non-Pioneer depth/movetime/game-clock control
- Selecting the game clock
- Starting through the explicit `START PLAYING THIS POSITION` action
- Starting from the edited position
- Cancelling setup and restoring the previous position

When setup is committed, the position is validated, Human versus Engine mode is restored, move history is cleared, and the edited position becomes the new starting FEN for saving and analysis. If the engine owns the selected side to move, its search starts immediately. Positions without exactly one king for each side or otherwise invalid board state remain in setup for correction.

Move Notation and History

The move-history panel displays standard algebraic notation rather than raw coordinate notation.

Examples include:

```
1. e4 c5
2. Nf3 Nc6
3. Bb5 a6
4. O-O Be7
```

The notation includes chess-specific forms such as captures, checks, checkmate, castling, and promotion. Internal UCI history is retained for engine communication and opening recognition.

The computer-status panel can display:

- Current side to move
- Opening name
- Selected engine level
- Displayed engine strength
- Engine control for non-Pioneer engines, for example `CONTROL FIXED DEPTH 6` or `CONTROL 2.5S / MOVE`
- Search depth
- Search time
- Node count
- Nodes per second
- Evaluation
- Opening-book status

Saving Games

Press SAVE to save the current game as a standard Portable Game Notation file with a `.pgn` extension.

Saved games include:

- Move history
- Standard algebraic notation generated by the chess model
- Custom setup starting positions
- FEN headers when the game did not start from the standard position
- Main-line move structure compatible with chess software

A game created from position setup can therefore be saved and reopened without losing its custom starting position.

Opening Games

Press OPEN and select a `.pgn` file.

Opening a game restores:

- The saved starting position
- The complete board position at the end of the game
- UCI move history
- Algebraic move notation
- Last move information
- Move count
- Game status

- Custom FEN setup information

The opened game is ready for replay navigation, analysis, or saving again.

Game Replay

When a saved game is open, the move list becomes interactive.

Replay Controls

- Click any move in the move list to jump to that position.
- The four keys under the move list: ■■ to the start, ■ one ply backward, ■ one ply forward, ■■ to the end. The move indicator shows the replayed move. They act during a replay and on a finished game (which they put into replay from its end); a game in progress is left alone.
- Use the Left Arrow key to move backward.
- Use the Right Arrow key to move forward.

Replay starts from the saved starting FEN and reconstructs the board by applying the selected number of moves. The full game history remains intact while browsing.

During replay, the status bar shows the current position, for example:

```
REPLAY 12/38
```

Board interaction is disabled while viewing replay positions so the saved game cannot be accidentally modified.

Engine Analysis

Press `ANALYZE` to analyze the current board position with the active UCI engine.

Analysis:

- Uses the current position without making a move.
- Shows the engine's best move.
- Updates depth, evaluation, nodes, time, and principal variation data when supplied by the engine.
- Works on standard games, opened PGNs, and custom setup positions.

The `HINT` command also asks the engine for a best move, but presents it as a hint rather than a full analysis workflow.

`MOVE NOW` immediately requests an engine move when the current mode allows one.

External UCI Engines

The `ENGINE` button opens a file chooser for another local chess engine executable.

Any engine that correctly implements the UCI protocol can be selected. Before switching, CZUBATRONIC validates the engine by sending:

```
uci
isready
quit
```

The engine must return both `uciok` and `readyok`. Invalid or incompatible files are rejected without replacing the currently active engine.

After a successful switch:

- The new engine becomes active immediately.
- The previous engine is closed cleanly.
- The status bar identifies the selected engine.
- Analysis, hints, engine moves, telemetry, and engine-vs-engine mode use the new engine.

CAPA96 is the bundled standard engine loaded when CZUBATRONIC starts. Pioneer remains available as a selectable alternative with its calibrated level ladder.

Remembered Standard Engine

The `ENGINE` dialog shows the current standard engine and offers `Set as standard engine (remember for next launch)`. The choice is written to `%LOCALAPPDATA%\CZUBATRONIC\engine_config.json` (`{"standard_engine_path": "..."}`). At startup the remembered path is validated: a missing file or a failed UCI handshake produces a warning and a fallback to the bundled CAPA96, so the application always starts. `RESET STANDARD ENGINE TO CAPA96` deletes the configuration file.

Opening Books

Press `BOOK` to choose an external opening-book file for the active engine. The chooser accepts common book formats such as `.bin`, `.txt`, and `.book`.

Opening books are engine-specific in the UCI protocol. CZUBATRONIC reads the engine's advertised UCI options and configures the book only when the engine exposes a compatible string option such as `BookFile` or `OpeningBook`. If an engine advertises only a check option such as `OwnBook`, it is using an internal or built-in book and cannot accept an external filename through the standard UCI interface. In that case the engine's own book remains under its control.

Piece Sets

The `PIECES` selector in the `OPTIONS` dialog currently provides:

- `2D GLOSSY`: glossy Staunton pieces with a dark outline, the set the program starts with
- `2D CURRENT`: the earlier CZUBATRONIC bitmap set
- `2D ORIGINAL`: the original production 2D piece set copied from the production application
- `2D WOOD`: modern wooden Staunton pieces
- `2D WOOD TRADITIONAL`: traditional wooden pieces

Changing the piece set:

- Clears cached piece images.
- Redraws the board immediately.
- Preserves the current board position and game state.
- Scales pieces with the board and fullscreen window.

Move Indicators

The `MOVE INDICATOR` selector in the `OPTIONS` dialog provides two visual modes.

LED Mode

- Selecting a piece lights the four grid crossings around the selected square: small discs of pure #FF0000 with a lighter centre, no glow, one per crossing however many marked squares share it (0.9.9; before, one LED at the lower right lay over a knight). They are drawn after all squares and pieces.

- After a move, the corner LEDs appear on both the departure and destination squares. `BOARD LEDS` in `OPTIONS` sets them `BRIGHT` or `DIM` (colour and glow scaled), remembered in `engine_config.json`.
- Completed-move LEDs disappear after three seconds.
- Engine moves also trigger the LED display.
- The LEDs are antialiased and cached at each board scale.

SQUARE Mode

Restores the traditional full-square move indication. The departure and destination squares receive the existing square highlight treatment instead of red LEDs.

Board Presentation

Board Flip

Press `FLIP` or the `B` key to reverse the board orientation. Flipping changes the visual orientation and coordinate labels without changing the actual chess position or side to move.

Responsive Scaling

The board automatically recalculates its square size when the window is resized or placed fullscreen. Pieces, LEDs, and procedural board textures scale with the live board dimensions.

Window Sizing and Display Scaling

The right-hand information panel is a fixed-size area: whatever does not fit is clipped, not scrolled. Since 0.9.1 the window therefore sizes itself from its content instead of a fixed 1100 x 680 pixels:

- At startup the application measures what the information panel needs (with the chess clock reserved even while hidden) and what the header row needs, opens the window at that size, and sets it as the minimum window size. Windows display scaling (for example 125 %) is covered automatically because the measurement uses the real font sizes. The height is capped at the screen height; on a Full HD screen at 125 % the panel needs slightly more than is available and the engine-output area gives way, while the move history keeps its minimum.
- `MOVE HISTORY` is laid out before `ENGINE OUTPUT`, so engine output shrinks and collapses first when space is short.
- The title `CZUBATRONIC 2400`, the fascia marking, and the `SYSTEM STATUS` bezel are sized from the measured fonts; the status bezel holds a caption plus a two-line status.
- The control buttons are units (a button, or a label with its menu) in two logical groups that wrap into as many rows as the window width allows: four rows at the standard width, two when maximised. When the number of rows changes, the minimum window height follows.

`tools\retro_layout_probe.py` verifies all of this from the source tree, optionally with a simulated scaling factor (`--scale 1.25`) and forced window states (`--geometry 900x560`, `zoomed`, `unzoom`); it saves a screenshot and exits non-zero when anything is clipped.

Panel Grips (Invisible)

Since 0.9.7 the console is drawn as one chassis without lines or handles between its parts, and the two layout grips are invisible. They are found with the mouse pointer alone:

- The gap between the board and the right-hand panel: moving the pointer slowly across it turns the pointer into a horizontal double arrow; press and drag to make the panel wider or narrower. The board gives way, the clock, the move history and the engine output grow with the panel.

- The gap between the display's keys and `MOVE HISTORY`: moving the pointer across it turns the pointer into a vertical double arrow; press and drag up or down to share the panel's height between the engine output and the move list.

Each grip is a few pixels wide, so the pointer has to cross it slowly. Both positions are stored in `engine_config.json` and restored at the next start. Support staff should point users who cannot find the grips to the double-arrow pointer; the end-user manual has a section "Finding the grips".

Procedural Board Texture

The board uses lightweight cached procedural rendering rather than image texture files for the squares. It provides:

- Warm creamy light squares
- Soft reddish-walnut dark squares
- Subtle low-contrast tonal variation
- Four deterministic texture variants per square type
- No animation
- No JavaScript
- No external texture assets

Piece Rendering

The application supports high-quality scaling of the active piece set while preserving the original 2D set's canvas format separately from the normalized current set.

Application Schemes

The `SCHEME` selector in the `OPTIONS` dialog provides four complete interface palettes:

- `WALNUT`: the original warm dedicated-computer finish
- `BLUE`: a cool blue electronic-instrument finish
- `GREEN`: a restrained green hardware finish
- `GRAPHITE`: dark neutral ABS-plastic gray with warm electronic displays

Changing scheme preserves the game, clock, selected pieces, and interaction state.

Branding

`assets\branding\dcs_chess_logo.png` (the DCS Chess wordmark, transparent PNG) is shown in the bottom-right corner of the control console, scaled to four text lines high so it follows display scaling. It is packed before the button rows, which wrap in the width left of it, so no control can ever sit on top of it.

Application Icon and Taskbar Identity

The application icon is a gold knight (from the bundled Chessmaster piece set) on a walnut square with a bronze rim. `tools\make_app_icon.py` renders it deterministically into `assets\icons\` as a multi-size `czubatronic.ico` (16 to 256 pixels) for the executable and the installer, and as PNGs that the application passes to Tk at runtime so the window, the taskbar and every dialog show the same icon. The application also registers its own Windows AppUserModelID at startup, so its taskbar button is never grouped with other Python windows when it is run from source.

Move Feedback and Controls

Audio Feedback

A short beep (a 1000 Hz tone of 90 ms) plays after a successfully executed human or engine move; a move that gives check gets two. The level is one of `OFF`, `QUIET`, `NORMAL` or `LOUD` (`OPTIONS`, `BEEP`), remembered in `engine_config.json`. The tones are generated by the program and played through the Windows sound system from a background thread, so the interface never waits; a machine without sound stays silent.

No beep is played when:

- A piece is merely selected
- An illegal move is attempted
- A hint is requested
- A position is analyzed
- The board is flipped

Available Controls

The controls sit in two logical groups (game controls, then engine and appearance settings) whose rows wrap to the window width, so every control is reachable at any window size.

- `NEW`: configure and start a new game
- `OPEN`: open a PGN game
- `SAVE`: save the current game as PGN
- `SETUP`: edit a position
- `TAKEBACK`: undo the latest move
- ■■ ■ ■ ■■ (the keys under the move list): to the start, one ply back, one ply forward, to the end during replay
- `FLIP`: change board orientation
- `HINT`: request an engine move suggestion
- `ANALYZE`: evaluate the current position
- `MOVE NOW`: request an immediate engine move
- `ENGINE`: select another UCI engine
- `ENGINE SETTINGS`: choose explicit fixed-depth, engine-movetime, or game-clock behavior for non-Pioneer engines
- `MATCH`: configure and run two independent UCI engines on the normal board
- `BOOK`: select a compatible external opening book
- `OPTIONS`: the dialog with `SCHEME`, `PIECES`, `MOVE INDICATOR`, `MODE` (Human vs Engine or Human vs Human), the `BEEP` volume (`OFF`, `QUIET`, `NORMAL`, `LOUD`) and the `ABOUT` button. These are not changed during play and took a row of the deck on small displays (readers' feedback, 2026-09-24). Scheme, piece set, indicator and beep level are stored in `engine_config.json` (`ui`), as are the window's size, position and maximised state (`ui.window`), restored at the next start and fitted to the screen.

Keyboard Shortcuts

Key	Action
`N`	Open New Game configuration
`T`	Take back the latest move

Key	Action
H	Request a hint
F	Move the engine immediately
B	Flip the board
Left Arrow	Previous replay position
Right Arrow	Next replay position

About and Impressum

The ABOUT dialog (behind OPTIONS) includes:

```
DCS Analytics & Digital Consultancy Services
Business Divisions of Universe Connect KG

Sachsenkorso 23
15834 Rangsdorf

Tel: 0049 175 979 73 08
Email: info@digitalconsultancysolutions.com

TEAM
Arne Baillière, creator of the CZUBATRONIC chess system
Thorsten Czub, author of Capa 9.6 / CapaBlanca
```

It also presents the creator statement, complete engine history, full Thorsten Czub interview with portrait, Story of Capa article, custom software services, copyright notice, license, warranty note, and distribution restrictions.

A Tribute to Computer Chess

CZUBATRONIC is a tribute to the chess programs of the 1980s and 1990s that gave so many players hours of fun. It is also a tribute to the programmers behind every generation of these machines and to the board-chess producers whose work made a lasting contribution to computer chess and computer history:

- Mephisto / Hegener + Glaser
- Fidelity Electronics
- Novag
- SciSys
- Saitek
- TASC
- CXG / Newcrest
- Applied Concepts / Chafitz
- Conchess
- Excalibur Electronics
- RadioShack / Tandy
- Milton Bradley
- Mattel Electronics
- Avalon Hill

- All board-chess producers and their programmers who helped advance the field

This tribute is part of the product's identity and is shown in the application's About section.

Supported File and Engine Formats

Piece Artwork Credits

The `2D WOOD` piece set is based on **Staunton-Pieces** by James Clarke (`clarkerubber`):

https://github.com/clarkerubber/Staunton-Pieces

The artwork is distributed under the MIT License. The complete license text is included with the piece assets in the application distribution.

Games

- Portable Game Notation (`.pgn`)
- Standard FEN positions embedded in PGN for custom setup games

Engines

- Local executable files accepted by the file chooser
- Must support the Universal Chess Interface protocol
- Must respond to `uciok` and `readyok`

Versioning and Release Notes

The product version lives in one place, `retro_chess_app\version.py` (`__version__`, for example `2026.1`). The release pipeline `installer\build_release.ps1` derives everything from it through `tools\stamp_version.py`: the installer filename and `AppVersion`, the executable's file properties (`installer\version_info.txt`, rendered from `version_info.template.txt`), the footers of these PDF manuals, and the version line in the `ABOUT` dialog.

Release notes live in `docs\CZUBATRONIC_RELEASE_NOTES.md`, newest version first, one `## Version ...` section per version; the build renders them to PDF alongside the manuals and ships the PDF in the installer, the portable ZIP and `release\Documentation\`. The build refuses to run when that file has no section for the current version, so an installer can never ship with stale notes.

Scope and Limitations

- The current desktop app is implemented with Tkinter.
- The current piece selector provides four 2D sets; there is no true 3D renderer.
- PGN replay follows the main line of the opened game.
- Engine analysis depends on the active engine's UCI output and capabilities.
- Pioneer alone uses calibrated CZUBATRONIC level profiles. Other engines use explicit depth, engine-movetime, or game-clock controls; arbitrary UCI option editing is not currently exposed.

License, Distribution, and Ownership

CZUBATRONIC 2400 is proprietary software, a DCS Chess product of Arne Baillière, trading as Universe Connect. The purchase grants a limited, non-exclusive right to use the program; the installer contains all runtime resources but no proprietary source code. Resale, redistribution, repackaging, rebranding, sublicensing, hosting, mirroring and commercial exploitation are prohibited without prior written permission, and only the rights holder or a party authorised in writing may publish official builds. complete End User License Agreement (`CZUBATRONIC_2400_LICENSE.txt`) is installed with the product.