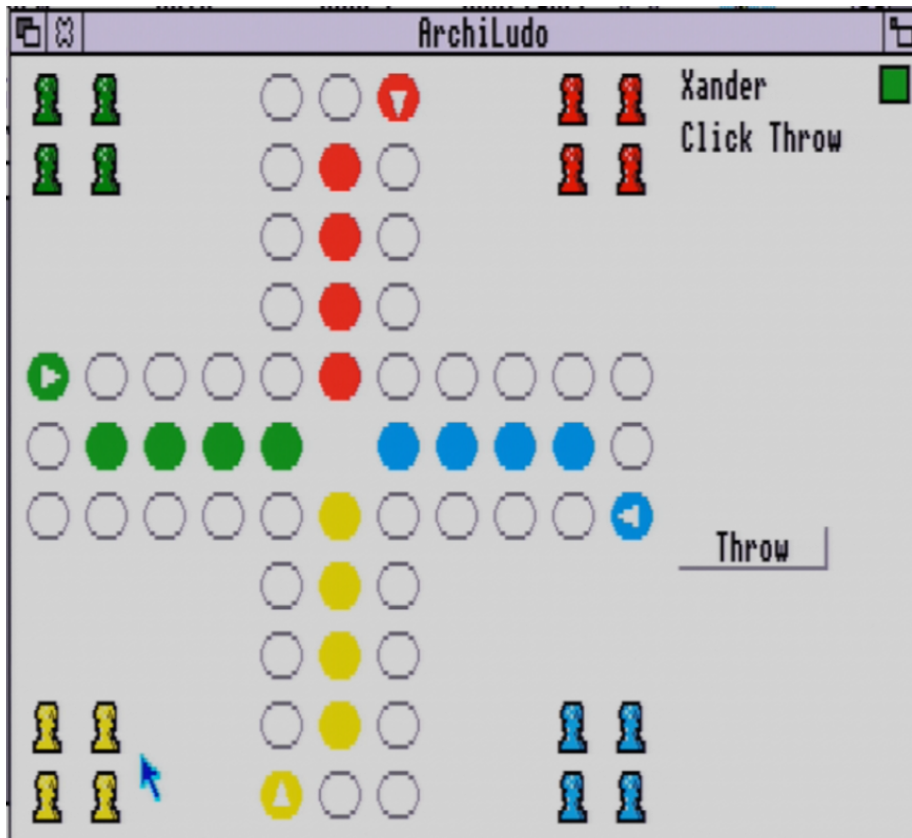


ArchiLudo



A Ludo (“Mens Erger Je Niet”) game for the Acorn Archimedes, targeting genuine 26-bit RISC OS 3.10. By Xander Mol (idreamtin8bits.com).

Contents

- [Playing ArchiLudo](#)
- [Interface manual](#)
- [Installation](#)
- [Building from source](#)
- [Changelog](#)
- [Development history](#)
- [Documentation](#)
- [Credits and licence](#)

Playing ArchiLudo

Four players, four pawns each, a shared ring and a short home run per player – the classic “Mens Erger Je Niet”/Ludo cross board.

Starting a game: on first launch, dismissing the startup splash screen (**OK**, or click anywhere on it) opens the **New Game** dialogue directly. Later on, click the iconbar icon to open it again. Set each of the 4 players’ name and whether they’re Human or AI-controlled; an AI-controlled player also gets a **Low/Medium/ High** difficulty picker (click to cycle) – see AI.md for what each tier actually does. Optionally click **Rules...** to pick a rule variant or adjust individual house rules (see RULES.md for the full manual), then **Start**.

Playing a turn: click **Throw** to roll. The status line tells you what to do next – click one of your own pawns on the board if more than one can legally move (if only one can, it moves automatically), or click **Throw** again on a six. Landing exactly on another pawn sends it home; reaching the exact end of your home column finishes that pawn. For an AI player’s turn, click **Continue** to step through its moves. Every time a player finishes, a win screen appears – **Continue** (offered for 1st, 2nd, and 3rd place) lets the remaining players keep going to decide the rest of the placing; the 4th and final player’s screen has no Continue, since nobody’s left to play. Every win screen also offers **New Game** and **Quit Game**.

Saving and loading: **Save Game/Load Game** from the iconbar menu open 5 fixed, renamable slots at any time during play.

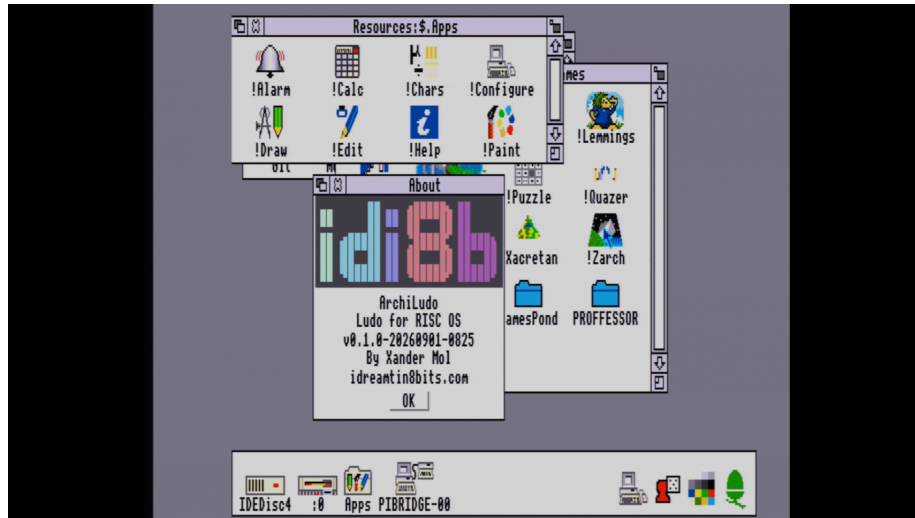
Music and sound effects: open the **Music** submenu (iconbar or window menu) to turn music **On/off**, turn **SFX** on/off independently, or pick a **Track**. See QTM.md for the full music/SFX manual (what each track and effect is, and Track 3’s one limitation).

Rules and variants: ArchiLudo ships 3 rule presets (Mens Erger Je Niet, Ludo, Pachisi-style) and 8 individually-adjustable house-rule toggles. See RULES.md for the complete rules manual.

Interface manual

All screenshots below are from a real Acorn A305 (ARM3, 4MB) over a PiEconet-Bridge, not an emulator – disc name PIBRIDGE-00 in the iconbar is the giveaway. This section walks every screen and dialogue in the order a player actually meets them; see “Playing ArchiLudo” above for the accompanying prose manual.

Splash screen





Shown once automatically on startup (and reachable again afterwards from the iconbar/window menu's **About** entry). Click **OK**, or anywhere on the window, to dismiss it – the very first time, this leads straight into the **New Game** dialogue below.

New Game dialogue

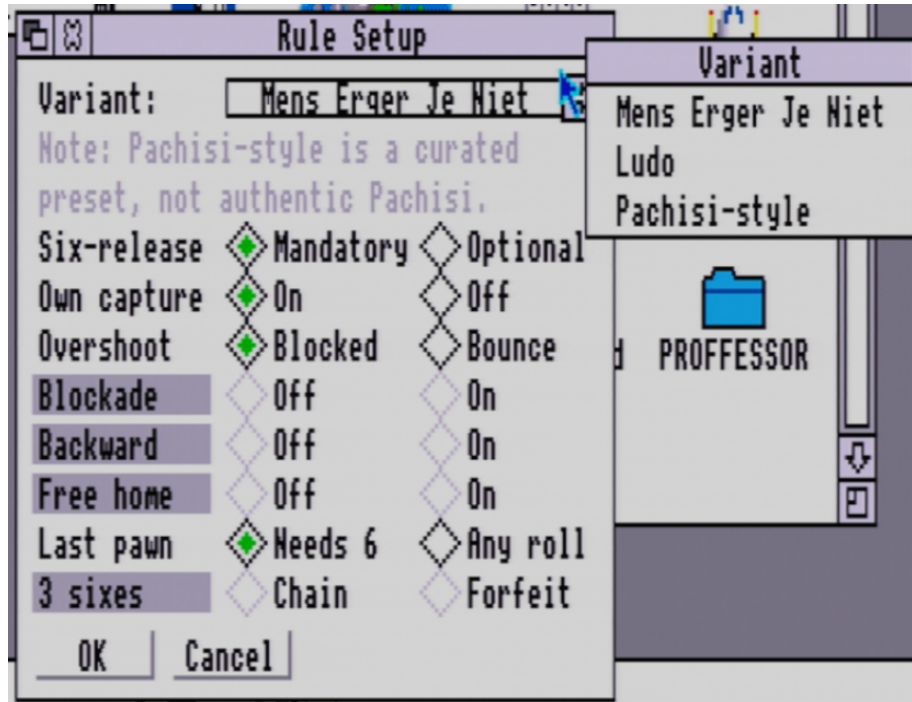


Set each of the 4 players' colour-coded name field and whether they're **Human** or **AI** (click to toggle). An AI-controlled player also gets a difficulty picker cycling **Low**/**Medium**/**High** (shaded out for a Human player, as GREEN's is here) – see AI.md for what each tier actually does.



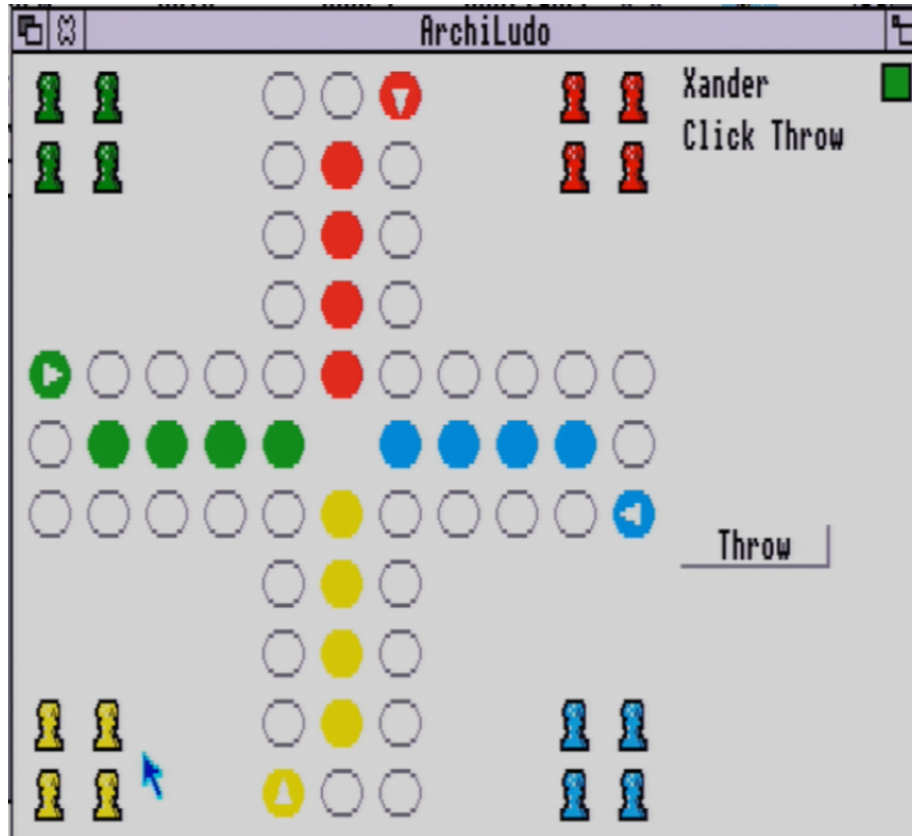
Any name field is freely editable (GREEN renamed to “Xander” here). **Rules...** opens the Rule Setup dialogue below; **Load** jumps straight to the Load Game dialogue instead of starting fresh; **Start** begins play with the configured players and whatever rules are currently selected.

Rule Setup dialogue



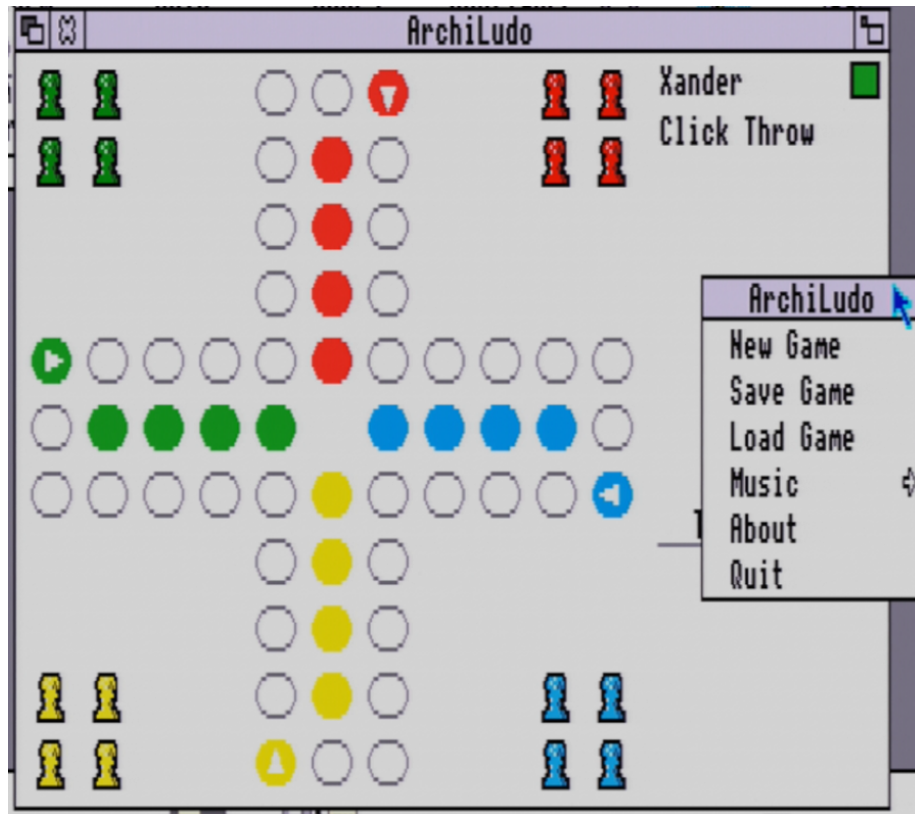
Reached via **Rules...** on the New Game dialogue. The **Variant** selector picks one of the 3 curated presets (Mens Erger Je Niet, Ludo, Pachisi-style) and fills in every toggle below to match it; any toggle can then be adjusted individually, which is what actually happens under the hood – a variant is just a starting point, not a locked mode. See RULES.md for exactly what each toggle changes about play.

The game window

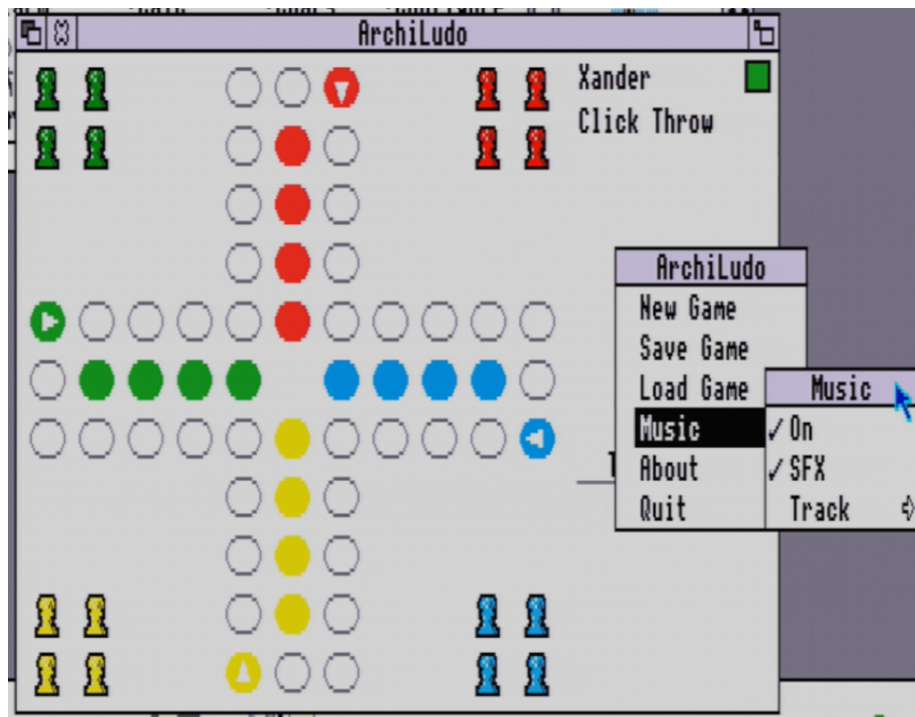


The board (also shown at the top of this README) fills most of the window: the shared 40-square ring, each player's 4-pawn home base in a corner, and their short home column running in from the ring to the centre. The panel on the right shows the current player's name and what to do next (**Click Throw**, or a prompt to pick a pawn), plus a solid colour swatch marking whose turn it is. Click **Throw** to roll; click one of your own pawns on the board to move it, if more than one legally can (a single legal pawn moves automatically without a click).

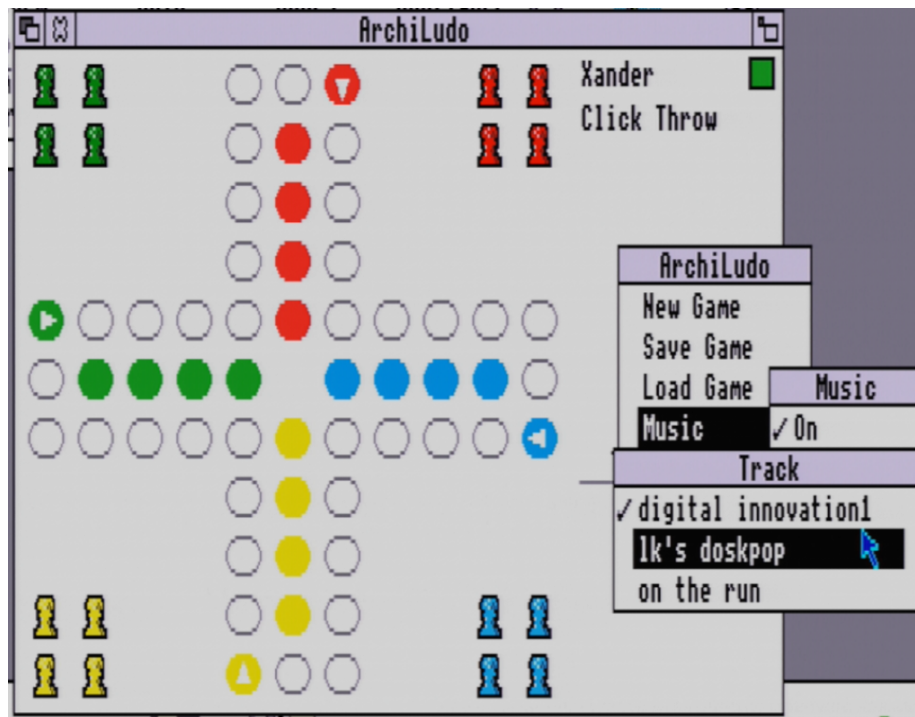
Menus



Opened from either the iconbar icon or a click on the game window itself (both show the same menu). **New Game** and **About** reopen the dialogues covered above; **Quit** closes ArchiLudo.

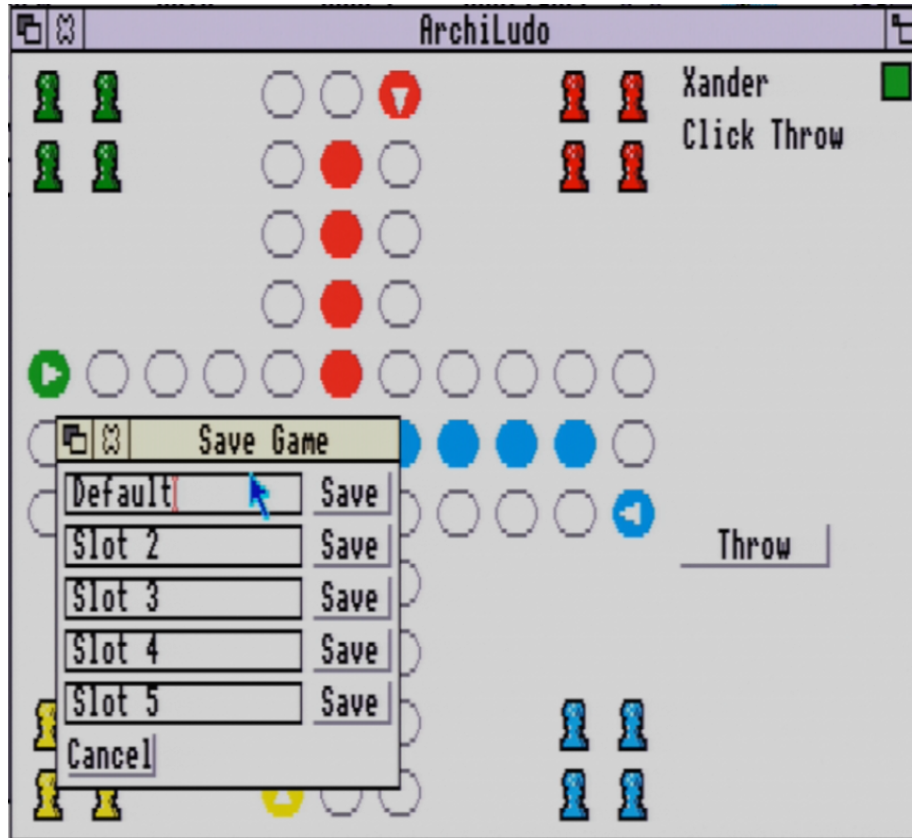


On and **SFX** are independent ticked toggles – music can play with effects muted, or vice versa. **Track** opens a further submenu (below) to pick which of the 3 bundled tracks plays.

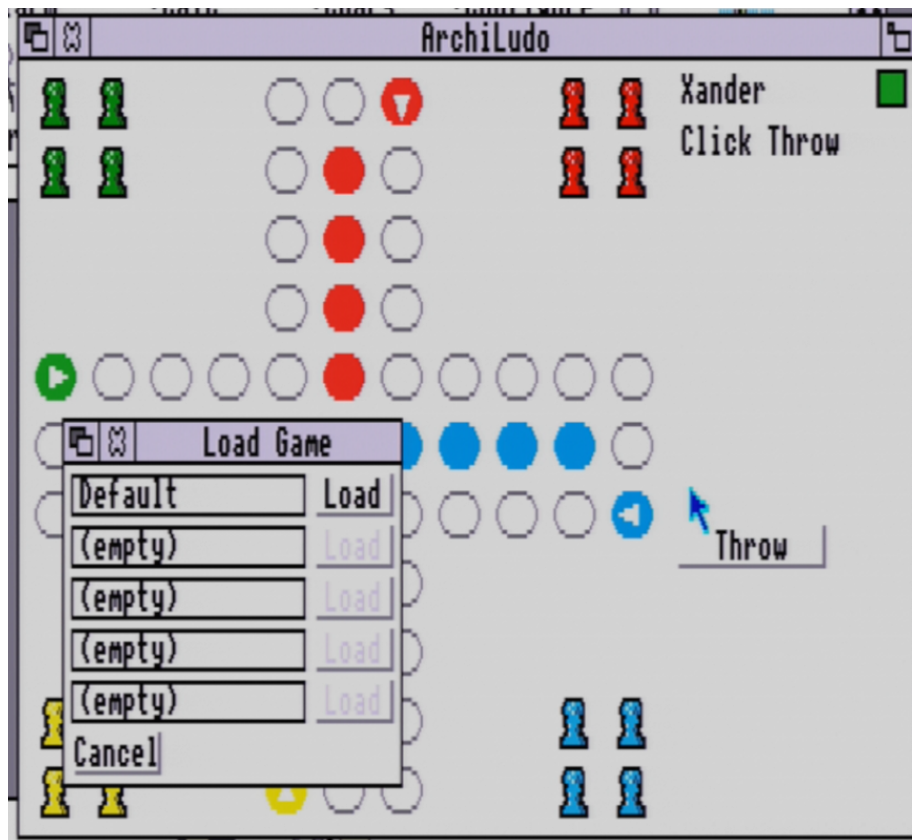


A tick marks the currently-playing track; clicking another switches to it immediately. See QTM.md for what each track and sound effect is, and Track 3's one limitation.

Saving and loading

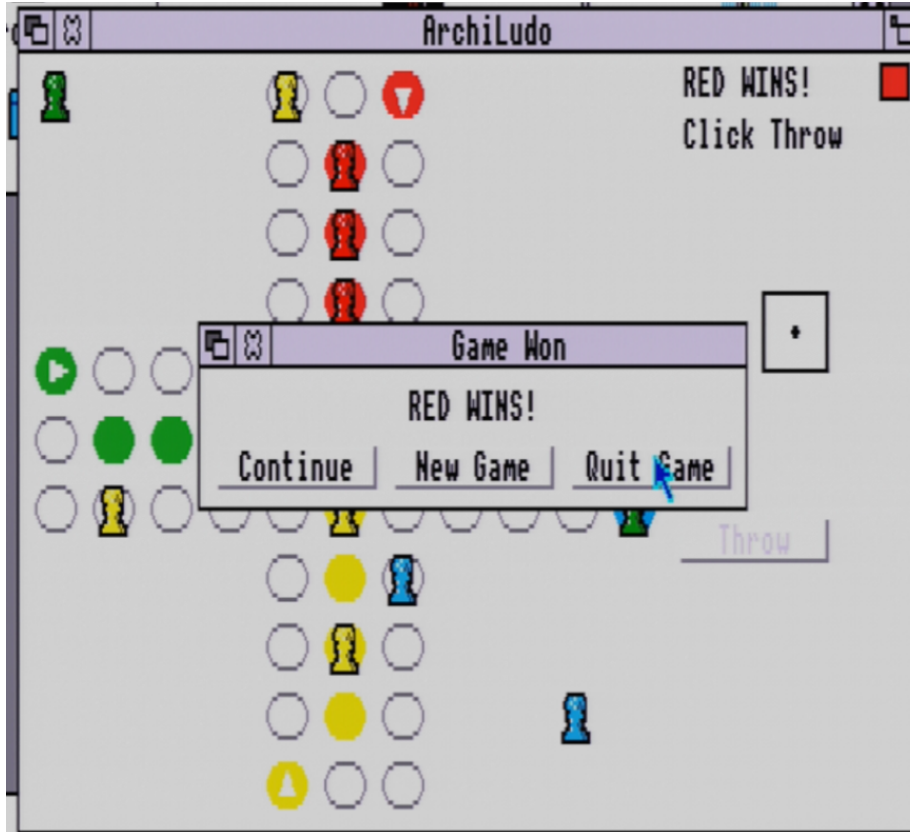


5 fixed slots, each independently renamable by editing its text field before clicking its own **Save** button – saving never prompts to overwrite, since all 5 slots are equally “yours” (no distinction between a fresh slot and one already in use).



An empty slot shows (empty) with its **Load** button shaded, exactly mirroring the Save dialogue's 5 slots. Reachable both from the iconbar/window menu and directly from the New Game dialogue's **Load** button.

Winning



A win screen (with fanfare) appears the moment any player finishes all 4 pawns, showing “<name> WINS!” for the first player home and “<name> ended 2nd/3rd/4th” for each place after that. **Continue** (offered for every place except the last) lets the remaining players keep going to decide the rest of the finishing order; every win screen also offers **New Game** and **Quit Game**.

Installation

Requires a real or emulated Acorn Archimedes running RISC OS 3.10 or later (ARM2 or ARM3). Download a release (ArchiLudo-vX.Y.Z-*.zip or .adf) and:

- **Zip**: extract with SparkFS (or any RISC-OS-aware zip tool) – filetypes are preserved. Double-click the extracted !ArchiLudo application directory to run.
- **Disc image** (.adf, ADFS “D” format, 800KB): write it to a real 3.5” floppy, or mount it directly in an emulator that supports raw ADFS images. It contains just the release zip, correctly filetype – extract it the same

way as above once it's accessible from RISC OS.

- **Real classic-Econet hardware** (a PiEconetBridge or similar old-style fileserver): rename the downloaded file to 10 characters or fewer with no dot before transferring it (e.g. **ArchiZip** for the zip, **ArchiADF** for the disc image) – such filesevers silently truncate longer names in a way that makes the file unreadable, not just renamed. Arculator's hostfs and a plain download/extract on Windows/Mac/Linux have no such limit.

No installer is needed beyond copying the **!ArchiLudo** directory wherever you want it – RISC OS applications are self-contained directories, not registry entries or system-wide installs.

Building from source

Prerequisites

Tool	Purpose	Install
ArchieSDK	ARM2-targeting cross-compiler (GCC 8.5.0), bundled OSLib	<code>git clone https://gitlab.com/_targz/archiesdk.git</code> <code>~/riscos-dev/archiesdk</code> <code>&& cd</code> <code>~/riscos-dev/archiesdk</code> <code>&& ./build.sh</code>
Arculator	Acorn Archimedes emulator, for testing	install anywhere; point <code>ARCULATOR_HOSTFS</code> in <code>.env</code> at its <code>hostfs</code> folder (e.g. <code>D:\Retro\Acorn\Arculator_V2.2_Windows\hostfs</code> on Windows)
pandoc	optional: README.md -> README.pdf	<code>sudo apt install pandoc texlive-xetex texlive-latex-extra</code>
Python 3 + Pillow	<code>tools/riscos_sprite.py</code> , the PNG->Sprite converter	<code>pip install Pillow</code>
sshpass	optional: <code>make deploy-pibridge</code> (real-hardware deploy)	<code>sudo apt install sshpass</code>

.env setup

Copy `.env.example` to `.env` and set:

```
ARCHIESDK = /home/xahmol/riscos-dev/archiesdk
ARCULATOR_HOSTFS = /mnt/d/Retro/Acorn/Arculator_V2.2_Windows/hostfs
```

PIBRIDGE_USER/PIBRIDGE_HOST/PIBRIDGE_PASS/PIBRIDGE_PATH are also needed for `make deploy-pibridge` (a real PiEconetBridge target) – see `.env.example` for the full set. `.env` is gitignored – never commit it.

`.env` is read only by `make` (a plain `-include .env` in the Makefile), not by your shell – if you use VS Code, `.vscode/c_cpp_properties.json` points its IntelliSense `compilerPath` at `${env:ARCHIESDK}`, which needs `ARCHIESDK` exported as a real environment variable separately (e.g. `export ARCHIESDK=/path/to/archiesdk` in your shell profile, or launch code `.` from a shell where you’ve already exported it).

Make targets

Target	Effect
<code>make / make all</code>	build the <code>build/!ArchiLudo</code> application directory
<code>make test</code>	build and run the game-logic unit tests with the host compiler (no ArchieSDK/Arculator needed)
<code>make clean</code>	remove <code>build/</code>
<code>make deploy</code>	copy <code>build/!ArchiLudo</code> to the Arculator hostfs folder
<code>make deploy-pibridge</code>	deploy to a real PiEconetBridge over SSH (password auth)
<code>make zip</code>	versioned release archive (<code>build/ArchiLudo-vX.Y.Z-<timestamp>.zip</code>), RISC OS filetype preserved
<code>make disk</code>	ADFS “D” format (800KB) disc image – the app and a plain-text README only (not <code>README.pdf</code> , too big to fit once it embeds every screenshot), correctly filetype
<code>make assets</code>	regenerate the pawn sprites and app icon from their Python generators
<code>make export-sprites / make import-sprites</code>	round-trip shipped sprites to hand-editable PNGs and back
<code>make docs</code>	regenerate <code>README.pdf</code> via pandoc
<code>make asm</code>	emit generated ARM assembly for the current sources into <code>build/</code>

See the “Installation” section above for the 10-character filename limit on real classic-Econet hardware – `make zip/make disk` deliberately keep the full version+timestamp in their own output filenames so multiple local builds can be told apart.

Testing: boot `configs/ArchiLudo-ARM3-4MB.cfg` (matches real ARM3/4MB hardware) or `configs/ArchiLudo-ARM2-1MB.cfg` (stock ARM2/1MB compatibility check) in Arculator with the RISC OS 3.10 ROM, then double-click !ArchiLudo from HostFS via the Filer (see BUILDCHAIN.md’s “Application directory” section for its structure).

Changelog

Release notes, newest first. See the GitHub releases page for downloads of any version.

v1.0.2

- **Sprite Pool crash fixed:** `tools/riscos_sprite.py`’s `cmd_pack()` wrote 0 as the “offset to next sprite” field for the LAST sprite in `!Sprites/!Sprites22`, as an end-of-area marker. Harmless when such a file is only ever read on its own – but merging a *second* application’s icon sprites into an already-populated Wimp Sprite Pool (via Filer’s own directory-listing icon lookup, or another app’s own `!Run’s IconSprites` command) then corrupts the pool’s own internal walk, throwing **Internal error: address exception** inside a ROM-resident sprite-pool-enumeration routine – confirmed live in Arculator, reproducible in both merge orders and with more than one app pair, and confirmed via several real, professionally-authored sprite files (checked against genuine 1990s Acorn apps) that the correct convention instead points that field at the sprite area’s own true end, even for the last sprite. Fixed, and `!Sprites/!Sprites22` regenerated – ArchiLudo now launches fine alongside another application in the same session, which it previously could not reliably do.

v1.0.1

- **Dice-roll fix:** `srand()` was never called anywhere – ArchieSDK’s `rand()` (a plain LCG, `SDK/src/libc/stdlib/rand.c`) starts from a hardcoded seed of 0 every launch, so every single game rolled the exact same dice sequence. Now seeded at startup and again when the New Game dialogue’s Start button is clicked, from `rand()`’s own current state mixed with `time(NULL)` and `os_read_monotonic_time()` – the second seeding point matters more in practice, since the time between launch and a player actually clicking Start is driven by human reaction time and differs every session, unlike the fairly fixed OS-boot-to-launch duration the startup seed alone captures.
- **Dice-parity fix:** that same LCG’s low-order bits are weak (an odd multiplier and odd increment make the low bit toggle on literally every call), so a plain `rand() % 6` made successive rolls’ parity alternate 100% of the time even though the 1-6 distribution looked uniform – verified by simulating the exact formula on the host over 600,000 rolls. Fixed

by using the healthier high-order bits instead. See `GAME_LOGIC.md`'s "Dice randomness and seeding" section for the full writeup.

- **GitHub social preview image:** a new `tools/gen_social_preview.py` composites a screenshot, the `idi8b` logo, and version/description text into the repo's social preview banner.

v1.0.0

First formal release – feature-complete, confirmed working both in Arculator and on a real, upgraded Acorn A305 (ARM3, 4MB) over a PiEconetBridge.

- **Documentation restructuring:** a new `TOOLS.md` manual now covers every standalone `tools/assets/*.py` script (usage, requirements, gotchas) in one place; `GRAPHICS_TOOLING.md`'s `sprite-format/rendering-approach` content moved into `ARCHITECTURE.md` instead of staying a separate file, and `BUILDCHAIN.md` now points to `TOOLS.md` rather than duplicating each script's own mechanics.
- **Cleanup:** removed dead code – a vestigial, never-actually-used sample-format conversion path in `lib/qtm.c` and the 6 one-shot SFX files it was the only reason to ship separately in the app directory (embedded into the music tracks' own MOD sample tables now, not shipped standalone); and a whole superseded pawn/dice art generator (`assets/generate_placeholder_art.py`) whose output nothing actually loaded any more, found while writing `TOOLS.md`. Debug file-logging (`debug_log()`) is now an opt-in build flag (`make DEBUG_LOG=1`) rather than always compiled in, so a default build carries none of its tracing strings or per-call file I/O.
- **Captured-pawn animation:** a displaced pawn (capture, own-pawn collision) now slides back to its home base instead of teleporting there instantly; the capture SFX was also replaced with a more expressive sound after two earlier tries read as too quiet or too flat in live testing – see `QTM.md`.
- **Full placement flow:** a win screen (with fanfare) now appears for every player who finishes, not just the first – "WINS!" for 1st, "ended 2nd/3rd/4th" for the rest, each offering Continue (except the last), New Game, and Quit Game.
- **AI difficulty tiers:** Low/Medium/High per AI-controlled player, picked in the New Game dialogue – Low takes only the objectively decisive moves (win/finish/capture/progress), Medium is the original full heuristic, High adds a genuine one-ply lookahead. See `AI.md`.
- **Launch flow:** the New Game dialogue now opens automatically right after dismissing the startup splash screen.
- **Distribution pipeline:** a filetype-preserving release zip (`make zip`), a from-scratch ADFS disc-image builder (`make disk`), and a real-hardware deploy target over Econet (`make deploy-pibridge`) – all verified against real hardware, not just Arculator, including finding and fixing a 10-character filename limit on classic-Econet filesystems and a line-ending

mismatch in the bundled plain-text README.

- **QTM audio:** background music (3 selectable tracks) and 6 one-shot sound effects, embedded as MOD instrument samples and triggered via `QTM_PlaySample`, independently toggleable, with loudness-normalized SFX. Confirmed working on real hardware.
- **Save/load rework:** from a PRM-correct but non-functional (on Arculator's HostFS) drag-and-drop design to 5 fixed, renamable save slots.
- **Multi-rule-set / house-rule variant system:** 3 curated presets (Mens Erger Je Niet, Ludo, Pachisi-style) plus 8 independently toggleable house rules, a Rule Options dialogue, and AI support for all of them – see `RULES.md`.
- **Win/continue flow:** a win no longer ends the game outright – players can continue for runner-up order or start a new game.
- **Sprite art pivot:** from `os_plot` primitives to real `Wimp_PlotIcon`-based pawn sprites and a proper application icon, plus a hand pixel-editing round-trip workflow for touching up art in an external editor.
- **AI opponents and player setup:** a “New Game” dialogue (names, Human/AI toggle per player) and a first AI opponent adapted from the original GeoLudo edition's own algorithm.
- **Animation and redraw overhaul:** flicker-free small-region animation (dice roll, pawn slide, pulsing highlights) via careful `Wimp_UpdateWindow` usage.
- **Application directory packaging:** a real `!ArchiLudo` app directory with a custom icon, replacing an earlier flat `hostfs` layout.
- **Phase 1: playable core loop:** the WIMP game window, the real Mens Erger Je Niet board (ported from the original GEOS edition's own coordinate tables), click-to-move, dice, capture/win detection.
- **Build environment:** the ArchieSDK toolchain, Arculator test profiles, and this project's full documentation set established.

Development history

ArchiLudo isn't this game's first outing – it's the latest in a line of ports going back over three decades, all by the same author. Full credits and technical detail for each generation are in `CREDITS.md`'s “Porting source” section; this is the narrative version.

- **1992:** the original, written in Commodore BASIC 7.0 for the Commodore 128 in 80-column mode – “Mens Erger Je Niet” (Dutch for “Don't Get Angry, Man”, the house-rule variant of Ludo this whole family of ports implements), by Xander Mol under the “XAMA Software” name. The listing itself still carries its original 1992 copyright banner. Source and a working D64 disk image are preserved at `C128/BASIC Original` in the `xamol/ludo` repository.
- **2020:** rewritten for the Oric Atmos, in BASIC.
- **2021:** rewritten a further three times – the Oric Atmos version again, now

in C via CC65; a new C port for the TI-99/4a via TMS9900-GCC; and a new C port for the Commodore 128 via CC65.

- **2023: GeoLudo**, a GEOS edition sharing one executable across GEOS on the Commodore 64, Commodore 128, and Commodore Plus/4 – `xahmo1/ludo/GEOS`. This is ArchiLudo’s own direct porting source: its board geometry and ring/home-column layout, its pawn and die-face artwork, and its first AI opponent all trace back to GeoLudo specifically (see `docs/ARCHITECTURE.md`’s GeoLudo-to-ArchiLudo mapping table), even though ArchiLudo’s own rules engine (`src/game_logic.c`) is a clean reimplementation rather than a line-for-line port (see `docs/GAME_LOGIC.md` for why).
- **ArchiLudo**: this project – a native WIMP port for the Acorn Archimedes, targeting genuine 26-bit RISC OS 3.10. The house rules the 1992 original encoded by hand are now the configurable `LUDO_VARIANT_MEJN` preset among three curated variants (see `RULES.md`), and its own multi-decade thread continues in this README’s Changelog above.

Documentation

See `docs/` for the full project documentation: `ARCHITECTURE.md` (layering, directory structure, porting notes), `RULES.md` (the full player-facing rules manual), `GAME_LOGIC.md` (rules engine API), `BOARD_LAYOUT.md` (board geometry), `AI.md` (AI design), `QTM.md` (music/SFX manual and audio API), `TOOLS.md` (every standalone asset/build tool, its usage, and gotchas), `BUILDCHAIN.md` (toolchain, Makefile, distribution), `OSLIB.md` and `LIBARCHIE.md` (the libraries this project builds against); plus `riscos_wimp_reference.md` for the WIMP/SWI/ message API reference, and `CREDITS.md` for everything this project is built on or derived from.

Credits and licence

See `CREDITS.md` for everything ArchiLudo is built on, ported from, or otherwise sourced from. Licensed under the GNU GPLv3.