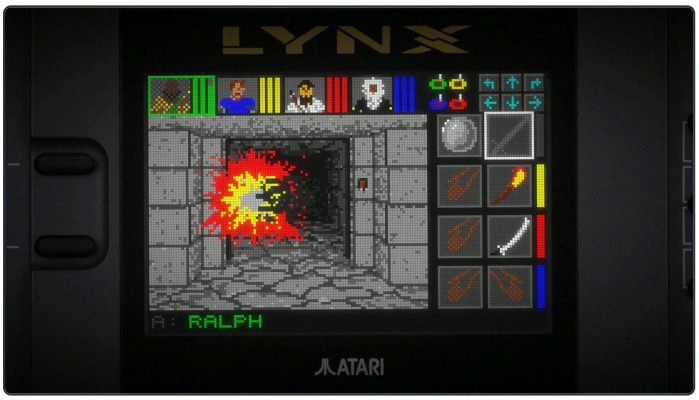
PROJECT SUMMARY · 28 OCTOBER 2026 · LYNX PORT V0.1, WORK IN PROGRESS

Dungeon Master for the Atari Lynx

FTL's 1987 Atari ST classic, reverse engineered from the original disk and rebuilt in 65C02 assembler for Atari's handheld



The Lynx version, recorded in the Felix emulator: the first mummy of the game, in the interface designed for the Lynx in October. Champions, formation and movement arrows on top, the 3D view, both hands of every champion on the right, one line of text.



From the second trailer: the game on a Lynx I, with the fine grid of the Lynx screen.

At a glance

Original	Dungeon Master, FTL Games, 1987 (Atari ST), version 1.2
Credits	Doug Bell (director), Dennis Walker, Mike Newton, Andy Jaros (graphics), Wayne Holder (producer)
Target	Atari Lynx, 65C02 at 4 MHz, 64 KB RAM, 160 × 102 pixels
Engine	Rebuilt from the reverse-engineered 68000 program, first in C, now entirely in 65C02 assembler
Assets	Read from the user's own disk image at build time (English or German v1.2)
Screen	3D view of 112 × 77 pixels at half the original scale; an interface designed for the Lynx with the information of the ST screens
Controls	Walk mode on the D-pad with a combat cursor, or a mouse mode with an accelerating pointer
Movement	Steps as in the original, or smooth: 5 to 8 in-between frames in real 3D
Sound	The original sound effects; stereo direction on the Lynx II
Saving	One saved game in the cartridge EEPROM (2 KB)
Cartridge	519,043 bytes (28 October); 15 code overlays share one RAM area
ROM Builder	Windows tool and Linux command line: build the Lynx version from your own disk image
Development	17–28 October 2026, by Claude in Claude Code; idea, design and play-testing by the project's author

Overview

This is a port of *Dungeon Master*, FTL's real-time dungeon role-playing game for the Atari ST, to the Atari Lynx handheld. The aim was the whole game, not a remake: the original dungeon with its 14 levels, the Hall of Champions with its 24 mirrors, the monsters and their behaviour, real-time combat, rune magic with all 25 spells,

the inventory, food and water, torches and darkness, the sounds, the intro with the entrance gate and the ending. The game logic follows the original routines. Graphics, the dungeon, texts and sounds are not redrawn or re-recorded: they are read from the user's own disk image every time the game is built. No original data is stored in the source code or in the tool.

Some additions go beyond the original. The most visible is smooth movement: instead of jumping from one square to the next, the view can glide there in real 3D, with five to eight in-between frames for every step and turn, and at every square the original picture appears again. Falling through a pit is animated the same way. The screens around the 3D view were redesigned for the small display and the D-pad: every champion's hands are always in view, a combat cursor lets the champions strike one after another with one button, and spells are cast in a small window over the dungeon. On a Lynx II the sounds come from the direction of their source, and a pause menu sets the movement mode and the volume and saves the game into the cartridge.

The **ROM Builder**, a Windows program with a command-line version for Linux, does the conversion for everybody who owns the game: it reads the disk image, converts the data and compiles the Lynx cartridge image on the user's PC.

The work was done by Claude, Anthropic's AI model, running in Claude Code on the author's Windows PC. The author set the goals, made the design decisions, tested every build and watched every video, and kept asking for more. Many features in this document began as the author's ideas, and many fixes as the author's observations; they are marked **AUTHOR** in the text. Claude did the reverse engineering, the programming, the tooling and the testing. The section *A.I. vs. human* near the end describes the division of the work in detail.

Tools

Reverse engineering	A Python toolchain written for this project: reader for Pasti (.stx) disk images, FAT12 file reader, unpacker for the packed game program, 68000 disassembler with a byte-for-byte round-trip check (reassembled with vasm), decoders for GRAPHICS.DAT and DUNGEON.DAT, and a Python reconstruction of the original 3D view. Ghidra 12.1.4 in headless mode, with a calling convention added for the Megamax C compiler, produced pseudo-C as a reading aid.
Lynx development	cc65 suite: ca65 assembler and ld65 linker (the C compiler built the versions until the switch to assembler); sim65 to run 6502 code on a simulated CPU.
Build & conversion	Python 3 without extra packages: the converter reads the disk image and writes the Lynx sprites, level records, sounds and texts.
Interface design	Python scripts that draw mockups of the Lynx screens from the game's own graphics and fonts – about 190 images in two days.
Testing & recording	"vlynx", a virtual Lynx built for this project (sim65 plus a Python model of Suzy's sprite drawing); a dozen differential test programs that run the C and the assembler version of a module side by side; the Felix Lynx emulator 0.6.4 with Lua scripts that log every displayed frame, every sound, every move of the party and the timing; ffmpeg for videos.
ROM Builder	PyInstaller: one .exe with Python, Tk and the needed part of cc65. For Linux a Python archive (.pyz) with cc65 for x86-64 and ARM64, compiled statically with zig and tested on a Raspberry Pi.

Trailers	Python with NumPy, SciPy and Pillow, ffmpeg; Blender 5.2 (Cycles) for the post-credits scene and the shots of the Lynx in the player's hands; public-domain photos of the Lynx I and II (Evan-Amos, Wikimedia Commons); the hands from a picture generated with Google Gemini; the fonts Cinzel and IM Fell English (SIL Open Font License); music written as code and played with the VSCO 2 Community Edition orchestra samples (CC0); one sound from Freesound (CC0).

How the work was done

1. **Understanding the original.** The game consists of a packed program with about 140 KB of 68000 code, GRAPHICS.DAT with 563 entries (images, sounds, fonts, tables) and DUNGEON.DAT with the 14 levels. Everything was read from the author's own disk. The author's first decision set the method: as in the earlier International Karate project, no external documentation and no fan references – everything had to come from the disk itself. The program was disassembled into a listing that reassembles to an identical binary, so names and comments could be added without risk. All 535 images decode exactly, all 485 routines of the game program got names, and seven Claude helper agents worked through the code in parallel. The result is a mechanics specification: the timeline that drives the world, sensors and doors, 27 monster types, champions and skills, 44 actions, 25 spells, the save format.

2. **A first view on the Lynx.** Before any game logic, the 3D view had to work. Claude built three mockups of the screen; the author chose the half-size view and a real Lynx with a flash cartridge as the target. Every fixed drawing call of the original became a pre-cut sprite at half resolution. The Lynx view was checked against the Python reconstruction for all 1,444 positions on the top level, and pixel by pixel in the emulator.

3. **The whole game.** Timeline, sensors, doors, teleporters and pits; monsters, combat and magic; the inventory, eating, drinking and sleeping; death and resurrection of champions; the end game. The author played every build and reported what was missing or wrong: the debug coordinates had to go, the mirror portraits were needed to choose champions, the champion boxes needed small portraits.

4. **Controls designed by the author.** The ST is played with a mouse; the Lynx has a D-pad and two buttons. The author wrote down a complete control scheme – a walk mode and a mouse mode switched with Option 1, weapon attacks with A held plus a direction, the spell panel enlarged on Option 2 – asked Claude to put the open questions one at a time, and answered them. The author also moved the heroes to the top and cut the messages to one line, as on the ST. **AUTHOR**

5. **Sound, intro and the English version.** "Is there no sound yet?" led to the sound player; the missing sounds the author noticed (eating, a portcullis going up) led to all of the original's sound calls being ported. The author wanted footsteps if the original had them – it does not, so the port has none. The author asked for the intro with the FTL logo, the title and the entrance gate, found the gate "simpler than the original" and the lettering hard to read, and suggested showing the credits scroll over the gate. The author also supplied the English disk, and the port has been built in English since the second day; the German disk still works. **AUTHOR**

6. **The ROM Builder.** The author's idea: the users bring their own disk, the project brings only the framework. Claude built the Windows tool and, at the author's request, removed the quoted game texts from its comments.

7. **Smooth movement.** The author asked whether the Lynx's sprite chip could make the step to the next square smooth, with the original graphics at every square. A first attempt with zoomed sprites "looked like a step back", then the floor stood still while the walls moved. The author proposed real textured 3D surfaces between the squares instead. A Python prototype was convincing, and the Lynx version followed (see below). **AUTHOR**

8. **Everything in assembler.** The C version used up the RAM. The author asked how much C was left and suggested converting all of it so that the C runtime library could go. Module by module, Claude and its helper agents rewrote the game in 65C02 assembler, each module checked against its C version in thousands of random scenarios.

9. **The pause menu.** The author specified it: the Pause button opens it, Pause with Option 1 restarts; it sets the movement (steps like the original or smooth in three qualities) and the volume, and saves and loads the game. The author also decided how saving works: into the 2 KB cartridge EEPROM, compressed, with a message when the game no longer fits. **AUTHOR**

10. **Polishing from videos.** After each milestone Claude recorded a demo video, and the author watched it frame by frame: colour flicker, wall decorations and a floor plate vanishing during movement, walls appearing for a moment where there are none, pictures on the wall staying rectangles while the view turns, an ornament disappearing too early next to the stairs. Each observation became a fix. When it turned out that the Felix emulator is closer to the real Lynx than the project's own virtual Lynx, the author asked for all recordings to be made in Felix. **AUTHOR**

11. **A larger window, against advice.** The author found too much empty space around the text line. Claude offered to stretch the view to 77 rows and advised against shrinking the original graphics anew from the ST – the stone joints would come out uneven and the pictures would need more RAM. The author wanted it tried anyway. It worked: the view is now 112 × 77 pixels, every piece resampled from the original. **AUTHOR**

12. **Falling through a pit.** "Very cool would be if falling into a pit were shown with in-between frames too" – now the view drops out of sight, the screen goes black, and the new level rises into view from above. **AUTHOR**

13. **Monsters and colours.** The author noticed that many monsters looked yellow, asked for every monster to be compared with the ST original, found a worm with a transparent mouth and a hooded creature without its glowing eyes – which must be exactly one pixel. The monsters now get the colours of their level as in the original. Then the author asked where one or two more colours would help most, which led to the palette split described below. **AUTHOR**

14. **Checking the magic.** The author asked whether all spells really work as in the original. An audit of all 25 spells against the original code found and fixed several deviations, among them explosions that never went away. It also brought the CAST button back to the original's behaviour; Claude recommended keeping it, and the author decided: like the original.

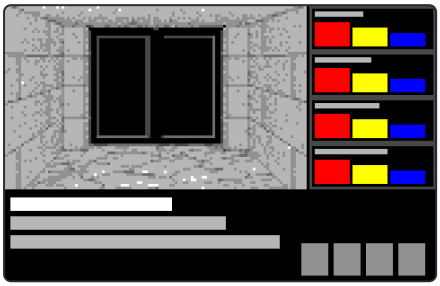
15. **A trailer, made in code.** The author asked for a professional trailer with Lynx and Atari logos, both Lynx models with the game on their screens, the pixel grid of the Lynx display, stereo sound, combat and magic explained and a creepy medieval atmosphere. Two days later the author added the idea for its end: a post-credits scene in which a mummy surprises the player. Both are described at the end of this document. **AUTHOR**

16. **A new interface for the Lynx.** On 26 October the author noticed that the longest champion name did not fit into one line of the candidate screen, and asked for designs that show everything the ST shows. Over two days Claude drew about 190 mockups from the game's own graphics, and the author chose and corrected them screen by screen: the candidate screen, the name entry, the inventory, the values behind the eye, and finally the main screen. The portraits stay on top, the spells move completely into a small window over the dungeon, the right panel shows both hands of every champion, and the six movement arrows of the ST came back into the top bar. The new interface was built on 27 October. **AUTHOR**

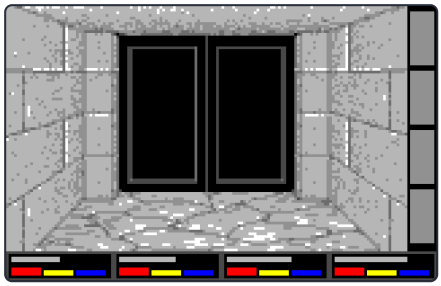
17. **Faster, leaner, darker.** The same day the author found "this short delay at the last frame before the stop" in a Felix video, asked whether memory could be saved by loading data just in time, how the mouse pointer accelerates, and whether the demon should not be black between its legs. Each question became a change: a quicker picture after every step, monster data streamed from the cartridge, a pointer driven by the vertical-blank interrupt, real black for the monsters. **AUTHOR**

18 **A second trailer and a teaser.** The author asked for a new trailer that shows only the new version and explains it: a bald champion reincarnated as a fencer, the inventory, eating, magic, bows, the combat cursor – then for a 30-second teaser with the mummy at the end, and for a poster as the first frame of both. **AUTHOR**

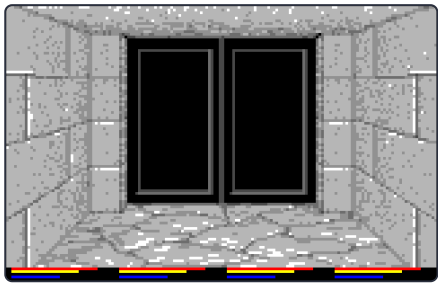
19 **Fixes from the trailer.** Watching the second trailer, the author saw that the movement arrows lit up only after the step, that the walls turned blue while the party fell through a pit, and that the delete key of the spell window was one pixel off. All three were fixed on 28 October and checked in Felix. **AUTHOR**



A: half-size view, 112 × 68



B: two-thirds, 149 × 91



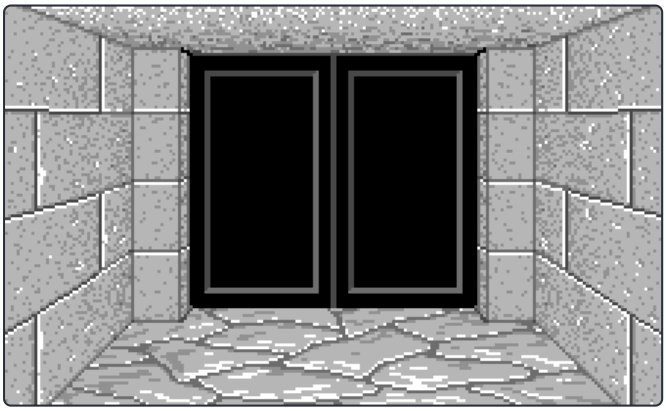
C: full width, 160 × 97

The first design question, on the first evening: three mockups of the same view, made from the original data. The author chose A. The next day the author moved the champions to the top and reduced the messages to one line, like the ST screen; on 22 October the view grew to 112 × 77, and on 27 October the interface around it was redesigned.

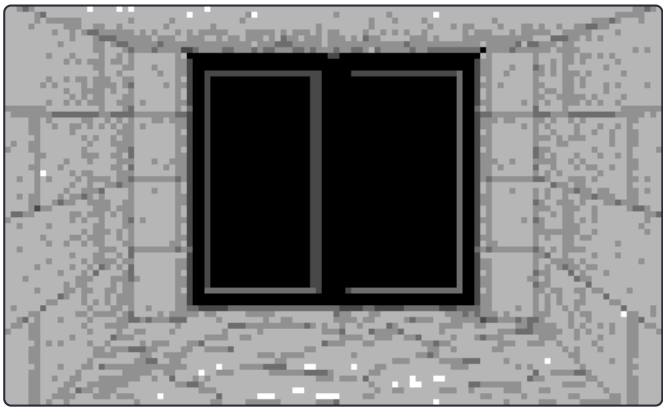
Technical challenges

Screen size: 224 × 136 into 112 × 77

The ST draws its 3D view in 224 × 136 pixels; the whole Lynx screen is 160 × 102. The view has to be halved, and halving destroys exactly what makes the dungeon look like stone: one-pixel joints between the blocks. The converter therefore works on every fixed drawing call of the original – a wall, a door frame, a staircase at a given position – and turns each into a ready-cut sprite, a “piece”, at the Lynx size. Area-weighted resampling decides each pixel; dark one-pixel joints survive, and the brightest colour only wins with a clear majority. Identical pieces are stored once, mirrored ones share data through Suzy's flip flag. The author noticed when distant walls lost their joints, and chose the version in which they stay visible. The intro uses its own rules: dot patterns of the gate survive as patterns, and the lettering is set at full resolution where halving would make it unreadable.



Atari ST: the original view, 224 × 136 pixels, drawn by the project's Python reconstruction of the ST renderer from the disk data.



Lynx: the same view assembled from pre-cut pieces at half size (first version, 112 × 68 pixels).

Memory: 64 KB, and the cartridge is not memory

The Lynx has 64 KB of RAM, and two screen buffers of 8,160 bytes each come out of that. The cartridge cannot be addressed like memory: it is read through an address counter that only counts forward, in blocks of 2 KB, so reaching a byte means reading through everything before it in its block. All graphics stay on the cartridge and are streamed into RAM when a view is drawn. Measurements in Felix showed where the time went: in the first versions a normal redraw took about 10 VBL, but drawing a whole view from RAM took less than one – the rest was loading overlays and graphics from the cartridge. Placing the most-drawn pieces at the start of their block cut the bytes read in vain by 96 %. The game code is split into 15 overlays – world, rendering, monsters, combat, magic, inventory, the 3D frames and others – that are loaded into one shared area as they are needed. The map of the dungeon does not stay in RAM either: a level is loaded from the cartridge when the party enters it, and changes to its squares are kept in a list that is applied whenever the level is loaded again.

RAM has been the scarcest resource from the start. At one point 2 bytes were free. Converting the program to assembler, replacing the C runtime and the graphics driver with own code and measuring the 6502 stack – in the measurements it never held more than 31 bytes, so a 144-byte buffer for the mouse pointer moved into the stack page – brought back enough room for the features that followed. In the last days the 702-byte table that describes the 27 monster types moved to the cartridge as well: a small cache in RAM holds the types of the current level, loaded when the level is entered. Today 1,120 bytes are free between the end of the program and the stack, and the 3D frames need at least 1,012 of them (see below).

The golden door and the last kilobyte

The author noticed that the golden ornament of a door disappeared while the party walked towards it. The in-between frames had not drawn door ornaments at all; once they did, the ornament still vanished now and then in the trailer recordings. The reason was memory: the ornament of a near door is a 504-byte block, and one of the places where the 3D frames keep such blocks is the free RAM between the end of the program and the stack. With too little free RAM the ornament no longer fitted. The stack measurement above freed the space, and since then every change is checked against that limit. When the author then asked whether the door also looks right while the view turns, Claude found a small jump in the door frame at the end of a turn; the author had it fixed in the game and left the finished trailer as it was. **AUTHOR**

Smooth movement in real 3D

The original jumps from square to square. The author's idea was to use Suzy, the Lynx's sprite chip, to glide between the squares, and the author set the rule that has guided all the work since: at every square, the original picture must appear again. The first approach scaled each piece of the old and the new view about the vanishing point. The author found it odd (“like a step back”), then saw the floor stand still while the walls moved, then side walls missing. The next morning the author proposed real textured surfaces instead.

The analysis showed that the original view is an exact pinhole camera: focal length 148 pixels, vanishing point at (56, 20), the camera 1.352 squares behind the centre of its square. With that model, a Python prototype drew the frames between two squares in 3D, and the author said: go ahead. On the Lynx, every visible wall face is drawn as narrow vertical strips of a texture made from the original wall picture; Suzy scales each strip to its height, which gives the perspective. Floor and ceiling are band sprites that Suzy stretches and tilts line by line. Doors, wall decorations, stairs and floor plates are flat planes in the scene; monsters and items are pictures that always face the camera, placed by the line of their feet. Suzy's hardware multiplier and divider do the maths. The last frame of every move is the original picture of the new square, drawn exactly as before.

Many details came from the author watching the result: the steep joints on near walls looked like staircases (now narrower strips, “I want the visually best solution”), the walls had different shades walking forward and back (now shaded by half steps), distant joints should become paler, the floor looked blotchy (now full width in four greys), pictures on the walls had to distort in perspective when turning. Asked whether pre-computed walls would be better, Claude calculated about 57 KB for the walls alone at three positions per move, against 9 KB that were free on the cartridge at that time, and no gain in speed. **AUTHOR**



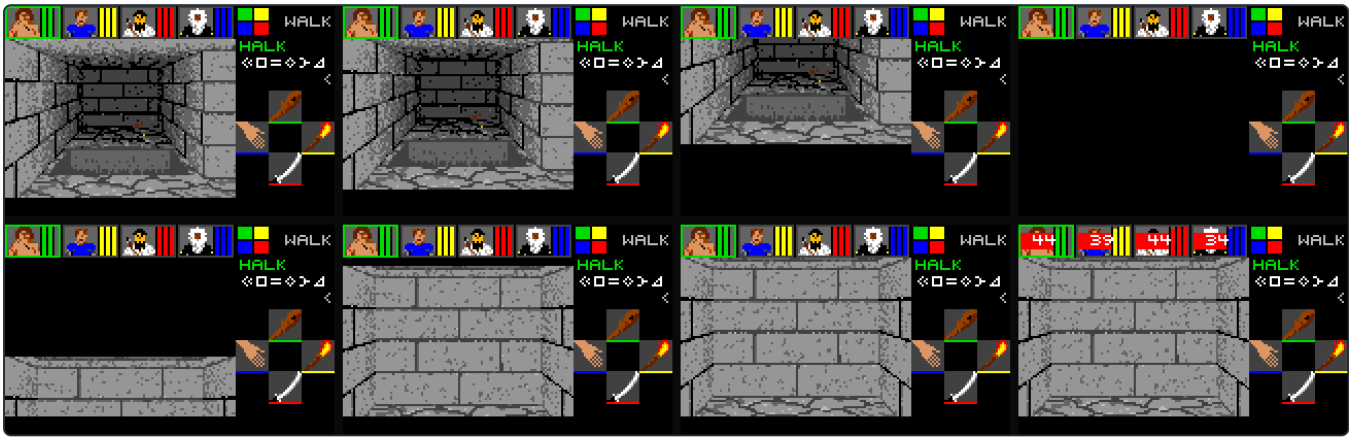
One step towards the golden door in the default SMOOTH mode, the 3D view only, from a Felix recording. Top left: the original picture; then six in-between frames drawn in real 3D, each shown for 3 to 4 VBL (60 to 80 ms); bottom right: the original picture of the new square. The door ornament moves with its door, and the joints are as thin as in the original pictures.

No flicker, no skipped frames

The author's rule for every animation: no flicker, and no frame may be skipped. Every in-between frame is drawn into the hidden screen buffer, shown at the vertical blank and kept on screen for a fixed time; when a frame is ready early, the game waits. The timing was measured in Felix with a Lua script that writes down when each frame appears and for how long. A frame of the fast mode must be finished within 2 VBL (40 ms at 50 Hz). When the walls were done, no frame took more than 1.9 VBL. Floor and ceiling cost Suzy less than 0.1 VBL of drawing; the walls needed 1.1 to 1.4 VBL, most of it calculation. The author then said they would rather give up frame rate than picture quality. Claude made comparison images and videos with 2 to 5 VBL per frame, and the author chose six frames of 3 VBL as the default: the SMOOTH mode shows six in-between frames for 3 VBL each, SMOOTH FAST eight for 2 VBL and SMOOTH FINE five for 4 VBL, with finer strips and more floor bands as the time allows. STEPS keeps the original movement. **AUTHOR**

Falling through a pit

The author's wish became its own overlay. The old view accelerates upwards out of the window, black follows from below, then the new level rises from the bottom as the camera sinks to eye level, settles with a small bounce and stands. Every frame is copied, not drawn: the CPU copies only about 3,000 bytes per VBL on the Lynx, so a frame may copy at most the 3D view, and each one stays on screen for 2 VBL. In the second trailer the author saw the walls of the old level turn blue as they fell away: the two palette entries that hold the monster colours (see below) already had the colours of the level below. Now the old level falls out with its own colours, and the new ones are switched in during the black frames, where nobody can see it. **AUTHOR**



Falling through a pit, recorded in Felix: the view slides up and out, the party falls through darkness, the level below rises into view, and the damage of the fall appears on the champions' bars.

Thin joints and the greys of the ST

The author noticed that the joints of near walls looked thick and blurred while moving and then jumped back to the thin lines of the original picture. The reason: the in-between frames took one texture from the front wall, halved in height to save memory. Claude showed the options as side-by-side videos, and the author chose the sharpest one: the textures now hold half a wall at full height, because the masonry repeats after two rows of stones, and every strip is drawn twice. Slanted surfaces get their own texture, unwarped from the original side wall. The author added that at every corner every second row of stones must end, and that the joints must look like the scaled original pictures, grey rather than black. The cost is time: in long corridors an in-between frame now takes 4 instead of 3 VBL.

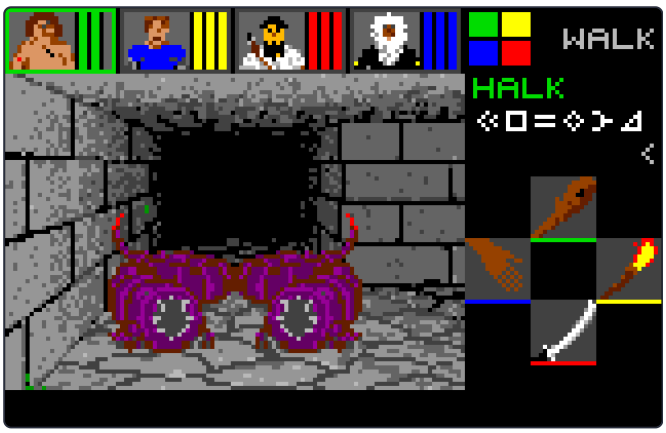
The black came from the light levels: in dim light the ST shows its darkest joint colour as a very dark grey that the Lynx palette did not have. “I think we need more dark greys”, the author said, and then: “Just switch to another palette when the monster is in view.” Palette entries 9 and 10 belong to the monsters of the level; when no monster that uses them is in the picture, the port now puts the two greys of the ST that the Lynx palette lacks on them, frame by frame. A test without the right-hand panel and with ten greys looked only slightly better, and the author kept the panel. The half steps of darkening in the in-between frames come from a random raster in the texture instead: two unused pen values mark every second stone pixel, which turns half a step darker sooner – no palette entry, no RAM, no time. The 56 bytes that the twice-drawn strips cost were found again elsewhere in the 3D code. **AUTHOR**

Colours: a palette split like the original

The ST and the Lynx both show 16 colours at a time. Dungeon Master uses all 16, and so does the port. In a sprite, a pixel is transparent when the sprite's pen table maps it to colour 0, so black needs care: pieces whose transparent pixels only lie at the ends of lines are drawn as opaque background sprites, and the pen tables darken the view for the six light levels. The original has one more trick: a raster interrupt changes two palette entries in the middle of the screen, so that every level can give its monsters their own colours. At first the port used the same two monster colours on every level, and the author saw “yellow” monsters and asked for a comparison with the ST. Since then every level has its monster colours again, but mapped to the nearest of the 16 colours in use. Then the author asked where one or two more colours would polish the look most. Claude compared and found the clearest gain in the worm on level 3 (purple in the original, red-brown on the Lynx) and the dragon on level 13. The port now does what the ST does: a timer interrupt switches palette entries 9 and 10 to the monster colours of the level for screen rows 16 to 92 and back again; the icons in the right panel are remapped so that they keep their colours. It costs little RAM and hardly any time. The author had first asked about line-by-line palette changes four days earlier. **AUTHOR**



Before: the worm of level 3 in the ST original (left) and in the port before the palette split (right), when it had to use the nearest of the 16 colours.



Now: the worms in purple on the Lynx, with two palette entries switched for the middle of the screen as on the ST. Their mouths are opaque black again since the author noticed that they were see-through.

Black for the monsters

Watching the demon in the second trailer, the author asked whether it should not be black between its legs: “I think we still lack a dark grey.” It should – but a monster is a sprite, and a sprite cannot draw colour 0, the black of the palette; that pen means “transparent”. The black parts of a monster therefore need one of the two monster colours to be black. Where neither of a level's two monster colours is black, the converter now looks for one that is almost the same as one of the 16 colours already in use (within a small distance). The monsters draw that colour with the colour in use instead, and the freed palette entry becomes black for that level – unless the level's walls or door ornaments use it. The demon, the dragon, the couatl, the trolins, the slime, the rats and several others got their black this way; for a few, among them the floating eye, the scorpion and the worm, it does not work out. In dark light levels a second rule applies: there entry 10 is black anyway. **AUTHOR**

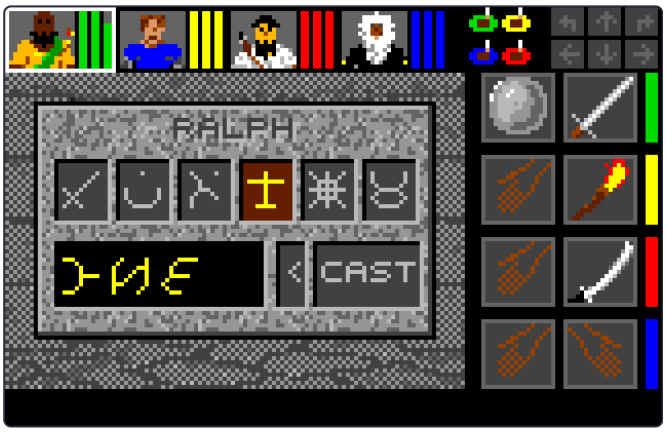
A new interface in 160 × 102 pixels

The first screen layout followed the ST's and squeezed it: the hands of the champions sat in a cross of weapon buttons, the spell panel took the right side, and some information of the ST screens was missing. The redesign kept the 3D view exactly as it was and rebuilt everything around it. The top bar shows the four portraits in the champions' colours, the party formation as small top-down figures that turn with the party, and the six movement arrows of the ST – thin, lit while a move is under way, greyed out outside walk mode, clickable in mouse mode. On the right, both hands of every champion sit in 17 × 17 boxes, flush with the top and bottom of the 3D view and with equal gaps (the author asked for both), each row with a bar that shows when the champion is ready again.

Combat needs speed on two buttons. A white frame, the combat cursor, marks the weapon hand of one champion: a short press on A lets that champion repeat the action he used last and moves the cursor to the next champion who is ready; holding A shows his actions in the text line, left and right choose one, releasing A executes it. Spells moved into a small window over the dungeon, a stone tablet with the runes drawn finely at double size: the top row chooses the caster, the second the runes, which light up when chosen, the bottom row holds the spell, a key to delete the last rune and CAST; outside the tablet the dungeon is dimmed with a checker pattern. The inventory shows everything the ST shows – all slots, food and water, the eye for the values, a lens to look at an item, chests and scrolls opened in place. The candidate screen in the Hall of Champions puts the whole name and title in one line. While designing it, Claude compared the ranks with screens rebuilt from the ST graphics and found that the port had shown every rank one step too high; it was corrected. The original's rule that a champion hit in close combat turns towards the strongest attacker had been missing as well and was added. **AUTHOR**



Design: one of the mockups, drawn by a Python script from the game's graphics: the main screen in walk mode with the combat cursor on the first champion.



Game: the spell window as built – a stone tablet over the dimmed dungeon, here with PAL FUL IR, a strong fireball, ready to cast.

Quicker pictures

Every change on the screen used to cost a full picture: the 3D view rebuilt from the cartridge, the interface overlay loaded and drawn – 12 to 16 VBL. When only the interface changes – a readiness bar grows, a message disappears – the port now copies the 3D view from the picture on the screen and redraws only the interface: 4 to 5 VBL. The author also found “this short delay at the last frame before the stop”: after the in-between frames, the full picture of the new square took 13 to 16 VBL, and the last in-between frame stood still meanwhile. Now the last in-between frame is already the new square in 3D, and the original picture of the square then appears first without the interface around it, which has not changed: 9 to 12 VBL. While a direction stays pressed, the next step starts at once, so continuous walking keeps its pace of about one square per second. **AUTHOR**

A pointer on a D-pad

In mouse mode the D-pad moves a pointer. The author asked whether it accelerates, so that short presses move it finely and holding it makes it run. It did, but its speed depended on how fast the game drew, and a short tap could be lost while a picture was being built. Now the vertical-blank interrupt counts the pointer's path 50 times a second, whatever the game is doing: a tap moves it by one pixel, after 6 frames it moves 2 pixels per frame, after 16 frames 3, and the game takes the counted path whenever it is ready. In Felix five of five short taps arrived, against two of five before. **AUTHOR**

The moment of the key press

In the second trailer the author noticed that the movement arrows in the top bar lit up about a second late. The arrow was drawn with the next full picture, and that came only after the in-between frames. Now the arrow is drawn straight into the picture on the screen at the moment the key is taken, the in-between frames carry it along, and it goes dark when the party arrives on the new square. In the original step mode, with no in-between frames, it stays lit for half a second after the press. If the pointer is over the arrow in mouse mode, it is taken off first and drawn again on top. **AUTHOR**

Suzy's quirks

The 3D frames use Suzy in ways the original pieces never did, and the emulator showed a few surprises. A sprite line stored in literal mode that ends exactly on a byte boundary is drawn wrongly, and a zero padding byte is drawn as pixels, so bit depths and strip widths were chosen to avoid both. Stretch and tilt values stay in Suzy for every later sprite that does not reload them, including the next normal view, so a tiny sprite resets them. Sprites that cross into the right panel are drawn with a shifted origin so that Suzy cuts them off at the panel. And for the CPU, the last rows of one screen buffer lie under Suzy's and Mikey's registers; copying there needs the memory map switched first.

Sound

The original plays 4-bit samples through the volume registers of the ST's sound chip. On the Lynx, a timer interrupt 5,495 times a second writes each sample through the original's volume table directly into the output of an audio channel. The samples stay on the cartridge; a 128-byte ring buffer is refilled between sprites. The original has 33 places in its code that start a sound, with rules for distance and priority; the port plays them at the same places. The author asked whether the Lynx II could play sounds from the direction they come from: now each sound sets the stereo attenuation of its channel by the angle to the party's view, and on a Lynx I the registers simply do nothing. The author decided to leave out a switch: it always tries stereo. Felix's own WAV recording did not reproduce sound written this way – the author heard only clicks in the first videos – so the recordings now log every sound start and rebuild the audio from the samples. **AUTHOR**

Saving into 2 KB

The original writes about 48 KB per saved game; the cartridge EEPROM holds 2 KB. The port saves only the difference from the state at the start of the game, which is mostly zeros, and packs it with a simple run-length code: a new game with four champions needs about 1.2 KB. A save is written in an order that makes an interrupted one invalid. If the game has grown too large, the menu says GAME TOO BIG TO SAVE and keeps the previous save – the behaviour the author chose. One trap: talking to the EEPROM moves the cartridge's address counter, and the cartridge reader, which remembers where it stopped, then read garbage. It first showed up as a crash in the intro.

Leaving C behind, without changing the game

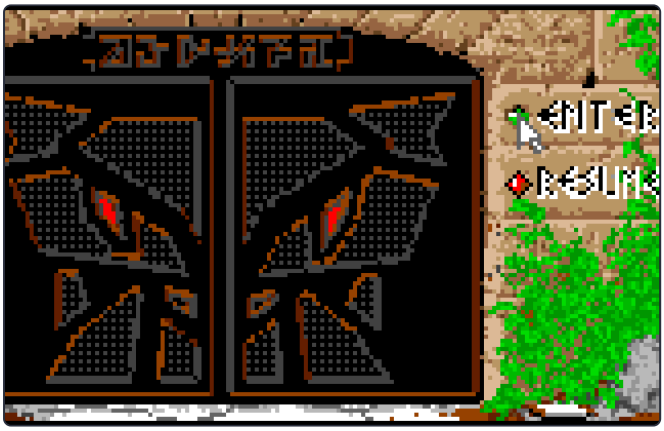
Every C module was rewritten in 65C02 assembler and kept as a reference. For each module a test program runs both versions side by side in sim65, with thousands of random game situations and with logs of every call and every random number; the results have to match exactly. Coverage was measured with sim65 traces (99 to 100 % of the instructions), and deliberately inserted errors were caught. The overlays shrank by a third or more – the user interface from 7,331 to 4,832 bytes, combat from 9,333 to 5,468, magic from 7,433 to 4,146. Finally the program got its own calling convention and start-up code: the Lynx build contains no C runtime and no C stack any more. The C references are still kept up to date: every later change, the whole new interface included, was made in both versions and compared again.

Testing without hardware

Three levels of checks. The view logic: sim65 builds the list of pieces for all 24,432 views of all 14 levels and compares them with the Python reconstruction of the original renderer. The game: complete scripted runs produce a checksum of the game state for every tick and of every frame, compared with reference runs after every change. The real thing: Felix recordings of every displayed frame, with timing and sound. The project's virtual Lynx was fast and convenient, but it had more free memory than a real Lynx and hid a problem that Felix showed at once – door parts and monsters vanishing during movement. So far everything has been tested in emulators; a test on a real Lynx with a flash cartridge is still to come.

The Lynx game today

Version 0.1 covers the game from the intro to the end game and is built from the English or German disk. The pictures were recorded in Felix.



The intro: the entrance gate with ENTER and RESUME, the lettering set at full height to stay readable.



The credits scroll over the gate, names fading in one after another – both the author's suggestions.



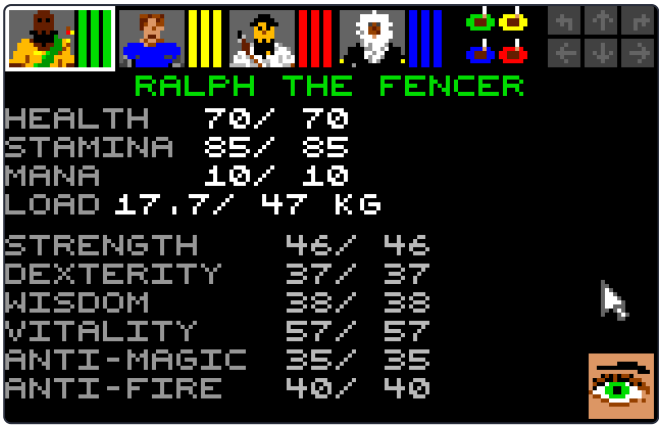
A candidate in the Hall of Champions: name and title in one line, classes and ranks beside the portrait, values in two columns, the equipment, and RESURRECT, REINCARNATE or CANCEL.



Reincarnation: a new name and title, typed with the characters of the original font.



The inventory with a chest opened in place: all slots of the original, food and water, the mouth, the eye and sleep.



Behind the eye: health, stamina, mana, load and the six values of the champion.



Combat: a champion slashes the mummy, and the star of the original shows the damage at the bottom right of the view.



Magic: a poison cloud, PAL OH VEN, on the mummy in the doorway.



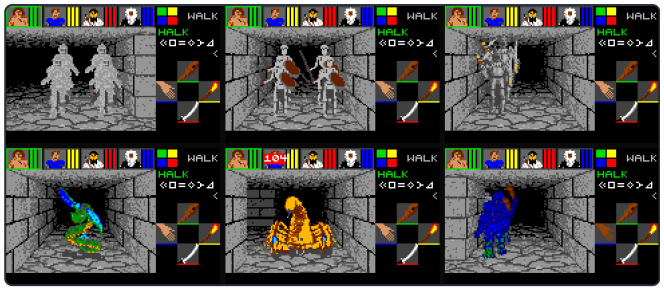
Bows and arrows: SHOOT, with the arrows left shown at the end of the text line.



A giant spider. CAN'T REACH: a champion in the back row cannot hit it with a sword.



The pause menu as the author specified it: movement, volume, saving and loading, and the version.



Some of the inhabitants: mummies, skeletons, an animated armour, a couatl, a giant scorpion and a trolin with its two blues.

What is in the game

- The original dungeon with all 14 levels, its doors, buttons, levers, keys and locks, pressure plates, pits, teleporters and illusory walls.
- The Hall of Champions with 24 candidates: resurrect them with their skills or reincarnate them with a new name and title.
- The monsters with their behaviour from the original: wandering, guarding, chasing, fleeing, smelling the party; monster generators, loot and thieves.
- Real-time combat with the weapon actions of the original, throwing and shooting; magic with the runes and all 25 spells, potions, scrolls and chests, food, water, poison and sleep.
- Light and darkness in six levels, torches that burn down, and spells such as light, see-through walls and footprints.
- The intro with the FTL logo, the title and the entrance gate, the credits, the game-over screen and the end game.
- An interface made for the Lynx: both hands of every champion always in view, a combat cursor for fast fights on one button, the spell window on Option 2, the inventory with values, item view, chests and scrolls; the damage star of the original.
- Two control modes: walk mode on the D-pad and a mouse mode with an accelerating pointer that can also jump between the click spots.
- Movement in four modes, chosen in the pause: STEPS like the original, or SMOOTH FAST, SMOOTH and SMOOTH FINE with in-between frames in real 3D; falls through pits are animated too.
- The original sound effects, with distance and priority as in the original, and their direction in stereo on the Lynx II.
- One saved game in the cartridge EEPROM (flash cartridges and emulators with EEPROM support); RESUME on the gate or LOAD GAME in the pause menu loads it.
- English or German game texts, depending on the disk.

485	24,432	8	190+	31
routines of the original program named and documented	views of all 14 levels checked against the Python reference	in-between frames per step in SMOOTH FAST (25 per second)	mockups drawn for the new interface	bytes: the deepest use of the 6502 stack that was measured

About 54,000 lines of 65C02 assembler (with comments) make up the Lynx game, 4,300 lines of Python the data converter, 4,000 lines the interface mockups, 13,600 lines the trailers and 1,500 lines the ROM Builder and the project page. Use the ROM Builder only with your own copy of the game and do not share the built .lnx file.

A.I. vs. human

This port would not exist without an AI, and it would not look like this without a human. The honest division is not "who typed the code" – the author typed none – but who decided what, and who noticed what.

Claude (AI)

- **Reverse engineering:** disassembling the 68000 program, decoding every file format, naming and documenting 485 routines, writing the mechanics specification.
- **Architecture:** pre-cut pieces, cartridge layout, overlays, the memory plan, the 3D camera model – the technical choices within the author's goals, each backed by measurements.
- **All code:** about 54,000 lines of 65C02 assembler, the data converter, build scripts, test programs, the virtual Lynx, the Felix scripts, the ROM Builder, the project page and the trailer pipeline. Helper agents worked on modules, sound design and music in parallel.
- **Testing:** differential tests, coverage, timing measurements in Felix, frame-by-frame comparisons, demo videos after each milestone.
- **Options:** mockups – about 190 for the new interface alone –, comparison images and videos, and a recommendation whenever a decision was due; explanations whenever the author asked how something works.

The author (human)

- **The project:** the idea, the goal (the complete game on a real Lynx), the method (only the own disk) and the privacy of the work.
- **The design:** screen layout, the control scheme, the pause menu, the new interface screen by screen, what the trailers should show and feel like.
- **The ideas that changed the port most:** using Suzy for movement, real 3D surfaces, more colours through palette changes, the animated pit fall, the ROM Builder approach, the post-credits mummy.
- **The quality bar:** no flicker and no skipped frames, the original picture at every square, faithful to the original where it is design.
- **The eye:** dozens of problems that no automated test caught – a step that looked like a step back, a phantom wall, wrong colours in a screenshot, an eye two pixels wide, a delete key one pixel off.
- **The choices:** picking between Claude's variants, overruling Claude when the author disagreed, and asking for a source when a claim had none.

In numbers: over twelve days the author wrote about 490 messages, more than 130 of them on 27 October, the day the new interface was designed and built. The longest was the control scheme; most were a sentence or two – "it looks like a step back", "the floor stands still and the walls move, that looks silly", "the eye may only have one pixel", "the heads twice don't look good", "the thumb presses up, not right". Claude answered with code, measurements, mockups and videos; the local repository holds more than 170 commits. Claude's first attempts were not always good: the first smooth movement looked "weird", its advice against the larger window proved too cautious, and the first hands for the post-credits scene looked "completely unrealistic". And some judgements stay human: Claude cannot listen to music the way a person does, so the trailer scores were composed as code and the author judged them by ear. Neither side would have made this port alone. The author put it this way in a forum:

> *"It is still a lot of work and thinking. Just not the classic coding."*

The author's ideas and decisions

The marks in the text above show where the author shaped the port. Here they are in order, taken from the project's conversation logs (the author's comments are quoted in translation).

DATE	THE AUTHOR'S INPUT	WHAT CAME OF IT
17 Oct	Reverse engineer the ST program first, "as with International Karate", without outside documentation; keep the project private.	The method of the whole project: everything derived from the own disk.

17 Oct	Chose the half-size view (layout A) and a real Lynx with a flash cartridge as the target.	The screen plan and the 512 KB cartridge budget.
18 Oct	Wrote the control scheme: walk and mouse mode on Option 1, A held plus a direction for the champions' attacks, released to strike; one text line; heroes on top.	The controls and the screen layout of the port.
18 Oct	Asked for sound and named the missing sounds (footsteps only if the original has them); asked for the intro, found the gate too plain and the lettering unreadable, suggested the credits scroll over the gate.	The original's sounds at its places, no invented footsteps; the intro as on the ST, readable at half resolution.
18 Oct	Supplied the English disk. “We only bring the framework”: users bring their own disk.	The English version; the ROM Builder, with no original data in the tool.
18 Oct	Use the Lynx's sprite chip for smooth steps; original graphics at every square.	The smooth-movement experiment, kept revertible on a branch.
19 Oct	“It all looks weird” – use real textured 3D surfaces between the squares.	The real-3D in-between frames.
19 Oct	Spotted the staircase effect on near walls, the missing joints on far walls, the blotchy floor; asked for the visually best solution.	Narrow wall strips, joints that fade with distance, a full-width floor.
19 Oct	“No flicker. No frame may be skipped.”	Double buffering with a fixed pace, verified in Felix.
19 Oct	Asked about line-by-line palette changes for more colours.	Groundwork for the palette split.
20 Oct	Convert all remaining C so that the runtime library goes; would rather lose frame rate than picture quality.	A game entirely in assembler; three smooth qualities, SMOOTH as default.
20 Oct	Specified the pause menu, saving and the “too big” message; renamed its items to LOAD GAME and BACK TO GAME.	The pause menu and EEPROM saves.
20 Oct	Found flicker, vanishing ornaments and floor plates, phantom walls, rectangles that should turn in perspective; asked for all recordings from Felix, which is closer to a real Lynx.	A round of fixes – ornaments, stairs and floor plates in 3D; a hidden memory problem found.
21 Oct	Asked for stereo from the direction of the sound, without a switch.	Stereo sound on the Lynx II.
22 Oct	Decided to fix a rare bug in the original's poison code that Claude had found (a champion strong enough to resist the poison completely could die of it).	Faithful where it is design, corrected where it is a bug.
22 Oct	Less space around the text, a larger window – “try the other variant anyway”.	The 112 × 77 view, against Claude's recommendation.
22 Oct	Show falling into a pit with in-between frames.	The pit-fall animation.
22 Oct	Yellow monsters, a see-through worm mouth, missing glowing eyes (“only one pixel”), the golden door ornament vanishing.	Level colours for monsters, opaque black, one-pixel highlights, door ornaments in 3D.
22–23 Oct	Do all spells really work as in the original? Then: keep the CAST button as in the original.	An audit of all 25 spells with several fixes; CAST works from the first rune, like on the ST.

23 Oct	Where would one or two more colours help most?	The palette split, as on the ST.
22–23 Oct	Trailer: the brief, and torches in wall holders like the game's, flickering, crackling, burning down to darkness.	The cold open and the atmosphere of the trailer.
23 Oct	Near joints thick and jumping while moving; every second row of stones must end at the corners; joints grey like the scaled originals, not black; “we need more dark greys” – switch palettes when a monster is in view; kept the right-hand panel.	Thin joints at full height, a side-wall texture, the missing ST greys on the monster slots, a random raster for half steps.
25 Oct	An end for the trailer: the Lynx in the player's hands, the torch almost burnt down, the handheld wobbling a little, a zoom out to the player's view, a mummy from the right – “Booh”.	The post-credits scene, rendered in Blender.
25 Oct	The mummy: eye holes completely black until the “Booh”, the mouth sewn up like a stitched wound that opens only a little, no thread ends through the cloth, arms that sit right at the shoulders, hands that look real.	A mummy modelled in code over many rounds; hands from a picture instead of a 3D model.
25–26 Oct	The Lynx in the end scene too grey, the hands glowing although there is almost no light; then a brightness chosen from a table of grades.	The Lynx as the original photo, only darker; hands below it.
25 Oct	A project page with the teaser, more information and pictures, but no download yet.	The project page on the author's website.
26 Oct	The longest champion name does not fit into one line – designs for the candidate screen, the inventory and the values page, with all the information of the ST version.	The new candidate screen, inventory and values page.
27 Oct	Main screen: keep the portraits; the spells completely in a small window over the dungeon; “the heads twice don't look good”; the six movement arrows of the ST, thinner, greyed out outside walk mode; hand boxes flush with the view and with equal gaps; finer runes that light up; a stone tablet with the dungeon dimmed around it.	The new interface, built the same day.
27 Oct	How does the ST show the damage done to a monster?	The original's damage star, chosen from several variants.
27 Oct	“There is still this short delay at the last frame before the stop”; can memory be saved by loading just in time?	The new square in the last in-between frame and a quicker picture after each step; the monster table on the cartridge.
27 Oct	Does the mouse pointer accelerate – fine with short presses, ever faster when held?	A pointer counted in the vertical-blank interrupt; no lost taps.
27 Oct	Should the demon not be black between its legs?	Real black for the monsters on most levels.
27 Oct	A new trailer with only the new version, explained step by step, a bald champion reincarnated as a fencer; a poster as the first frame; a 30-second teaser with the mummy at the end.	The second trailer and the teaser.

27 Oct	Second trailer: the music of the first one when the champions are chosen, the fencer last and calmly, the fall through the pit with the scream, the golden door only once, more variety, only smooth movement after the switch, no dragon and no demon (“we don't want to show everything yet”), one real zoom in and one zoom out with the hands.	The final cut of the second trailer.
27 Oct	The hands in the zoom-out should press the keys that match the game on the screen – “the thumb presses up, not right”.	Thumbs driven by the move log of the recording.
28 Oct	The movement arrows light up far too late; the delete key's sign should move one pixel left; the lines are blue during the fall through the pit.	Arrows lit from the key press until the party arrives, a centred delete key, the right colours while falling.

The trailers

Three videos were made without a video editor, in 4K and 1080p at 50 frames per second with stereo sound. The game scenes were recorded in Felix with scripted input; everything else – the cut on a fixed beat grid, the Lynx's pixel grid, the photos of both Lynx models with the game on their screens, the titles, the runes in the game's own font, a rebuilt Dungeon Master lettering at the end – is rendered frame by frame in Python. The music was composed as code for orchestra samples; a sound library of about 80 effects holds trailer versions of the game's own samples and synthesised atmosphere.

The first trailer, about two and a half minutes long, shows the port before the new interface and is on YouTube (youtu.be/vdJmbibAb1I). The author directed it like a film. The device and its screen wobbled against each other until the author suggested rendering them as one unit first and moving that. The blacks of the screen became darker, the consoles darker to suit the gloomy setting, the full-screen scenes moved onto a Lynx I with a floating camera. The demo party grew to four champions; the big dragon was left out in favour of the spider and the mummies; a lightning bolt was dropped because, as in the original, it looks almost like a fireball. And the opening is the author's: a wall of the dungeon, two torches in wall holders modelled on the game's own, flickering, crackling, then burning down quickly until darkness falls. **AUTHOR**

After the credits comes the author's post-credits scene, the only part made in 3D: a Lynx in the player's hands in a dark dungeon, the game on its screen, the torch almost burnt down. The camera pulls back to the player's eyes, a mummy shuffles in from the right, the player turns his head – and the mummy shouts “Booh”. It was built in Blender 5.2 with Python scripts. The author shaped the mummy over many rounds: eye holes that stay completely black until the “Booh” and then show two points of light, a mouth sewn shut like a stitched wound whose threads give way only a little, arms that sit right at the shoulders. Hands modelled in 3D never looked real, so the hands come from a picture the author generated with Google Gemini, cut out and laid onto the public-domain photo of the Lynx. The “Booh” is a CC0 recording from Freesound. **AUTHOR**

The second trailer, three and a half minutes, shows only the version with the new interface and explains it step by step: the champions chosen in the Hall to the beat of the music, a reincarnation as a fencer, the inventory with eating and drinking, the runes, real-time combat with the combat cursor, bows, secrets, the switch from steps to smooth movement. It has one deep zoom into the Lynx screen and one zoom out from the screen to the Lynx in the player's hands, rendered in Blender: there the thumbs press the keys that match the game on the screen, read from the log of the moves in the recording. The second trailer and the teaser start with a poster as their first frame. The 30-second teaser is pure action on a faster beat and ends, as the author asked, with the mummy. **AUTHOR**



The opening of the first trailer: torches in wall holders like those of the game, shortly before they go out.



The post-credits scene: the mummy, rendered in Blender, a moment before the “Booh”.



The second trailer: the zoom out from the screen ends on the Lynx in the player's hands, the thumbs on the keys.



The poster, the first frame of the second trailer and the teaser.

Current status

Version 0.1 runs in the Felix emulator. Still open: a test on a real Lynx with a flash cartridge (including the stereo channels and mirrored sprites) and saved games late in the game, which may not fit into the 2 KB EEPROM. The ROM Builder, for Windows and as a command line for Linux on x86-64 and ARM64, builds the game from the English or German disk image and checks the result against the reference build of the port; it is not published yet.

Dungeon Master © 1987 FTL Games / Software Heaven, Inc.; the notice in the game reads “Copyright (c) 1987, Software Heaven, Inc.” Lynx port: unofficial, non-commercial fan project, developed with Claude Code; not affiliated with or endorsed by the rights holders. The reverse engineering used only the author's own copy of the game and no outside documentation. Atari and Lynx are trademarks of Atari. Screenshots: recorded inside the Felix emulator from the Lynx build; the original view and the layout mockups drawn by the project's tools from the disk data; Lynx photos: Evan-Amos, public domain; hands in the trailer scenes: generated with Google Gemini. The author's comments are quoted in translation from German.