RELICS

RELICS: The Market as Medium

A Fully On-Chain Artwork Bound to a Uniswap v4 Pool

> Live on Ethereum mainnet since 3 August 2026. Every factual claim below is a public read against chain id 1; the addresses are in Appendix E.

VERSION **2.0.0**

PUBLICATION PHASE **MAINNET_LIVE**

SOURCE COMMIT **9127deba840a**

CANONICAL **relics.wtf/whitepaper**

CONTENTS

Abstract

A Uniswap v4 pool can call an external contract during its own lifecycle — when it is initialized, when liquidity is added, when a swap settles. RELICS uses that programmability as an artistic medium rather than a financial one. It is a fully on-chain generative artwork: a finite civilization of up to ten thousand artifacts, each drawn by a contract from Ethereum's state at the moment it is read, with no stored image and no external service. Every artifact has a permanent identity, fixed once in a word of on-chain DNA, and a mutable appearance recomputed from that identity together with a bounded record of the market that trades it. A hook contract, bound to one canonical pool, keeps that record — directional volume, price extremes, drawdown, volatility — and can only observe it: under its deployed permissions it cannot alter, tax, or refuse a trade. The renderer translates the record into visual material: fracture, repair, sediment, light. The result is an artwork that is permanent but never finished. Its rules cannot change; its state can. It does not change because time passes. It changes because the market that trades it makes history.

1. The market as medium

Generative art begins with a seed and an algorithm. A number is drawn, a procedure runs, and an image results — one fixed output, or a bounded family of them. The procedure is the work; the seed decides which member of the family you receive. Everything that will ever be true of the image is settled the moment the seed is chosen.

RELICS keeps the seed and the procedure and adds a third input: the recorded history of the market in which the work is traded. That history is not a decoration bolted to the artwork from outside. The pool where RELICS changes hands is part of the environment the artwork runs in, and the events that happen there — trades settling, liquidity arriving, a price reaching a level it has never reached before — are made legible to the renderer that draws each artifact. The image is a function of the identity *and* of what the market has done.

A price feed tells an artwork a number. This is different in kind. The market does not merely assign a price to RELICS; it leaves marks on it. A sustained sell-off darkens the work and reddens its wounds. A recovery lets some of that darkness lift. A new high is recorded permanently, as a tier the work has reached and cannot un-reach. The pressure is public and irreversible, and the artwork keeps it.

None of this romanticizes the market. The system records what happens whether the trading is euphoric, destructive, dead quiet, coordinated, or adversarial. It does not know which, and it does not judge. It keeps a bounded record of collective behavior and hands that record to a renderer, and the renderer makes it visible. What the market means is left to whoever is looking.

2. Permanent and unfinished

The reason to put an artwork on a chain is not storage. It is permanence of a particular kind, and to see which kind, it helps to separate three things that are usually run together.

The first is **a file stored permanently** — an image held somewhere durable. Most work sold as on-chain art is this at best, and usually less: a token points at a server, the server returns a picture, and the picture lasts as long as someone pays to host it. The chain holds a receipt; someone else holds the art.

The second is **a procedure preserved permanently** — the rules kept intact, so the work can be executed again from them. This is the older tradition of conceptual and generative art. Sol LeWitt's wall drawings are instructions, closer to a musical score than to a painting; the work is the certificate and the diagram, and a crew executes it on a wall.[1] Vera Molnár wrote programs for machines that were retired decades ago.[2] Casey Reas has argued for years that in this kind of work the rules are the artwork and the image is one performance of them.[3] When the work is a procedure, the procedure has to survive, and procedures rot faster than paint — they depend on hardware, on institutions, on someone caring enough to migrate them before the last machine dies.[4] Larva Labs' Autoglyphs answered this cleanly: the generator is embedded in the contract, and each glyph is computed and then fixed at the moment it is minted — created on the chain rather than merely registered there.[5] The procedure is permanent and its output is permanent. The work is finished.

RELICS takes the third condition: **a procedure preserved permanently and left exposed to future state.** The rules are fixed on Ethereum and run themselves, with no curator and no migration path. The identity of each artifact is fixed. But the image the rules produce is not settled, because part of what the rules read is a record that the market keeps writing. On-chain artwork that responds to live chain data is not unprecedented — Art Blocks has shipped work whose parameters remain writable on chain after mint.[6] RELICS's one distinct move, claimed without any suggestion of being first or better, is to *narrow* the state it is exposed to: not arbitrary chain data, but a bounded history of one market, captured by a contract attached to that market's own pool.

This is why immutability and change are not opposites here. The rules cannot change. The state can. A finished object cannot change at all; a Relic simply has not changed yet. That is the paradox the rest of this paper is about.

3. The pool as a living environment

The market RELICS listens to is one specific canonical pool: $RELICS against WETH on Uniswap v4, bound to the artwork's hook before that pool existed and fixed there permanently. This specificity matters. RELICS does not read an arbitrary price feed that could be pointed anywhere; it reads the one pool it was married to, and it will read no other.

Treat that pool as four things at once. It is a **market**, where the work is exchanged. It is a **public stage**, where that exchange happens in the open and cannot be taken back. It is a **sensor**, because the hook attached to it turns the pool's own operations into recorded state. And it is a **historical environment** — the weather system the artifacts live inside, accumulating a record of real swaps, real liquidity, real movement in price, real distance fallen below a former high, and real recovery back toward it.

Because the pool is real and singular, its history cannot be faked cheaply or borrowed from elsewhere. It can be *moved* — anyone with enough capital can push a price, and Section 11 is candid about that — but it cannot be counterfeited, and the record is kept from inside the pool's execution rather than reconstructed after the fact by a third party who might get it wrong. Market state here is artistic input, not a trusted financial oracle. The artwork never uses it to decide what anyone may buy, sell, or withdraw. It uses it only to decide how the work looks.

---

4. The hook as an artistic instrument

This is the mechanism that makes everything above possible, so it is worth stating plainly what a hook is.

Uniswap v4 lets a pool carry an attached contract, a **hook**, that the protocol's central `PoolManager` calls at defined moments in the pool's life. Which moments a given hook is allowed to intervene in is encoded in the hook's own address; a pool's operations call only the permissions its hook actually holds. Earlier automated markets could be *read* from the outside, after the fact. v4 lets a contract be invoked from *inside* the pool's own operations, at the instant they happen.

The RELICS hook holds exactly three permissions, and they are all observational:

POOL EVENT	WHAT THE HOOK OBSERVES	WHAT IT RECORDS	ARTISTIC CONSEQUENCE
Pool initialized (`afterInitialize`)	The opening price and tick	Seeds the starting tick, the all-time-high tick, and the first market seed; refuses any opening price other than the one it was pre-committed to	Fixes the origin of the work's history — the zero point every later mark is measured from
Liquidity added (`afterAddLiquidity`)	The size of the addition	Increments a liquidity-event count and a cumulative liquidity magnitude	Structural ground: how much depth the civilization stands on
Swap settled (`afterSwap`)	Direction and size of the trade, and the resulting tick	Buy and sell volume, net flow, the latest tick, any new all-time high, the drawdown beneath it, and running stress and volatility	The pressure that becomes fracture, repair, instability, and historical tier

Everything else is switched off. There is no callback before a swap, no callback when liquidity is *removed*, and none of the permissions that would let a hook return a price adjustment or claim a fee. The consequence is exact and verifiable: **the RELICS hook cannot alter a trade, take a fee, or refuse one.** It is a recording instrument wired to three moments, deaf to everything else. (One honest

limit follows directly: the hook sees liquidity arrive but never leave, so its record measures how much liquidity has been added, not how much remains. A sparse-liquidity reading means little was ever added, not that liquidity drained away.)

That restraint is what makes the hook an *instrument* rather than a mechanism. An artist chose precisely what it would hear and what it would ignore, wrote those choices into a contract, and then gave up the ability to change them. The hook's single job is to turn the liquidity pool from an external market the artwork could only look at into an internal input the artwork is built from.

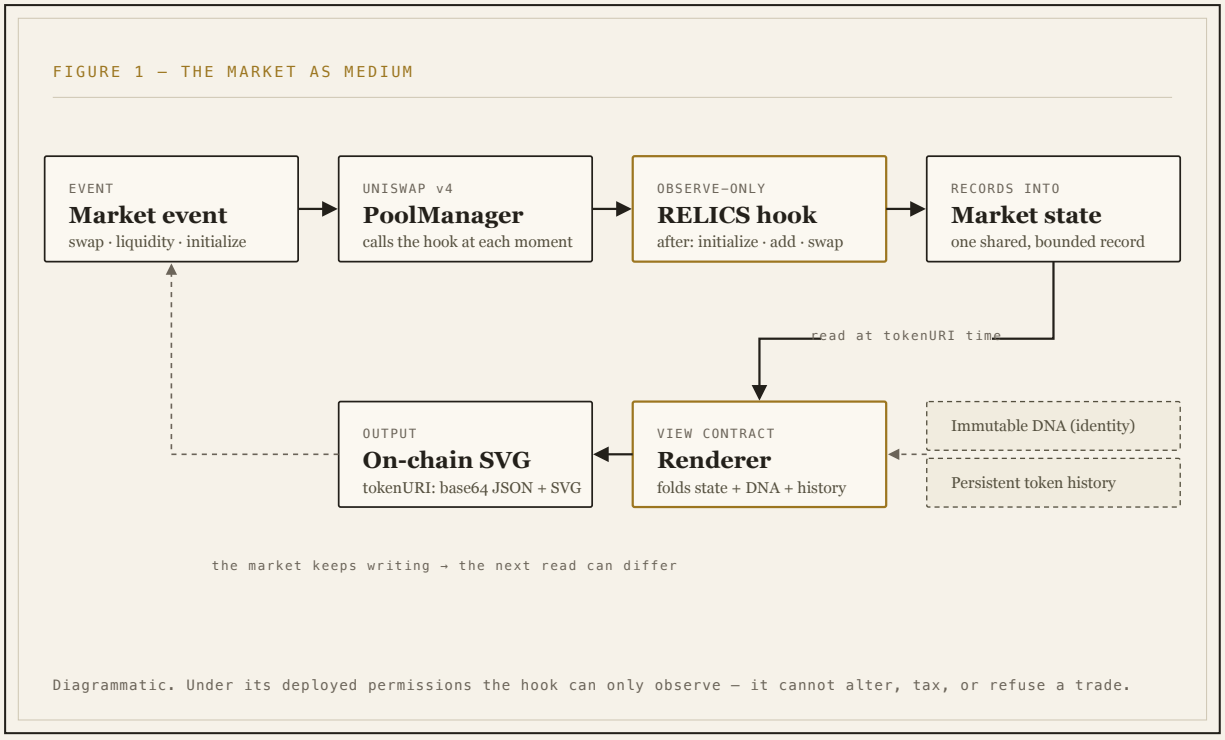


Figure 1. How a market event becomes visual form. A swap or liquidity change is reported by the PoolManager to the hook, which records it into one shared market state; at read time the renderer folds that state, the Relic's DNA, and its token history into the on-chain SVG.

5. Immutable DNA, mutable phenotype

Each Relic is computed from three things:

> **immutable DNA + persistent token history + shared market state = the current Relic.**

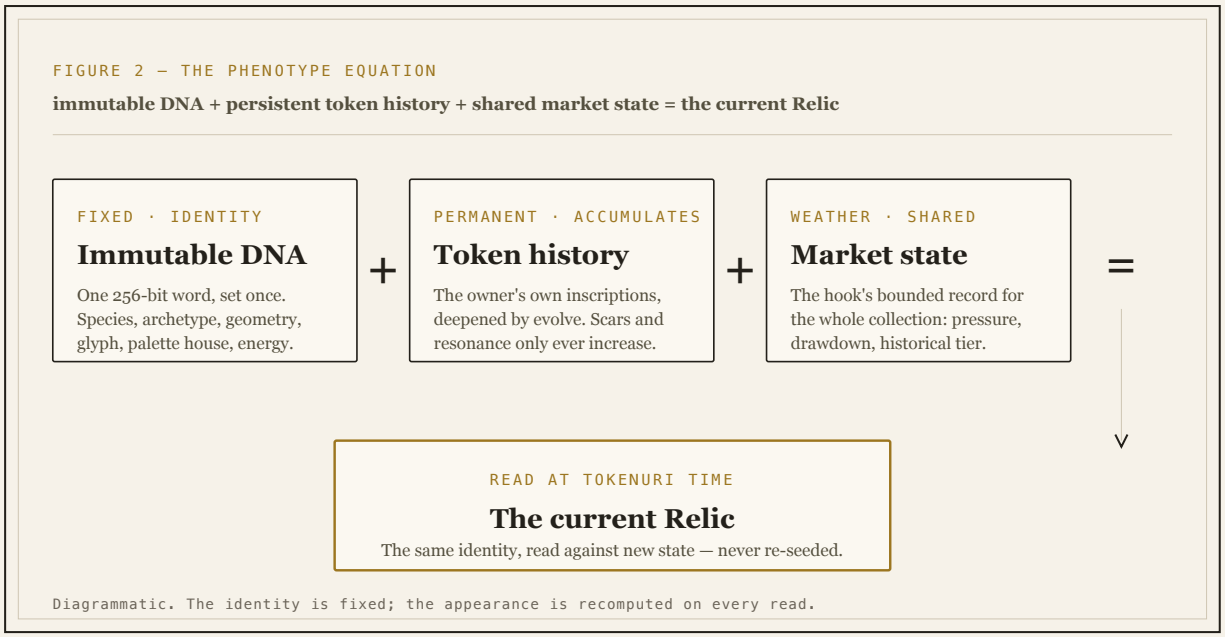


Figure 2. The three inputs to every read — immutable DNA (identity), persistent token history (owner inscriptions), and shared market state (the hook's record) — resolve to one current Relic.

DNA is the identity: a single 256-bit word generated once, in the transaction that first brings a Relic into existence, and thereafter only read. From it come six attributes that never move — its species, its archetype, its geometry class, its glyph family, its palette house, and its energy signature. Colour is drawn from a different part of the word than form, so a vessel shape recurs across unrelated palettes and a palette is recognisable across unrelated shapes, the way real material cultures work rather than the way a trait stack does. These six are who the Relic is. No later event touches them.

Shared market state is the hook's record, described in Section 4. It is one bounded structure for the whole collection — no per-Relic data, and the hook never loops over Relics — so every artifact reads the same market history. This is what makes the work respond to the world: corruption, historical tier, liquidity exposure, volatility exposure and recovery are recomputed from it on every read.

Persistent token history is what an individual Relic has permanently accumulated: an owner may call a function that reads the live market and deepens their own Relic's scarring and resonance, and those values only ever increase. This is the one place a single holder inscribes a single object, and it cannot be undone.

The distinction between the three inputs produces a **phenotype**, the current appearance, with three different temporal behaviours, and the honesty of the work depends on keeping them apart. Some properties are **fixed**: the six identity attributes. Some are **permanent record that only accumulates**: the owner's inscriptions, and the all-time-high tick the market has reached, which ratchets up and never down. And some are **current weather**: the drawdown, stress and volatility that rise when the market falls and recede when it recovers. A deep collective drawdown darkens the ground and reddens the rendered wounds of every awakened Relic on its next read; as the market climbs back toward its high, that particular darkness lifts. The geology stays; the weather passes.

Across all of it the Relic remains itself. It is never re-seeded and never reissued. It is the same identity read against new state — which is exactly why its appearance can change without its identity changing at all.

6. One civilization, ten thousand bodies

RELICS is not ten thousand unrelated pictures that happen to be sold together. It is one civilization of up to ten thousand identities, and what makes it one is the single market history they all share.

The archetypes are recurring institutional forms — an Abyssal Idol, a Ritual Machine, a Sacred Shard, a Living Archive, a Buried Monolith, a Containment Core — crossed with palette houses that read as distinct material cultures. Each body is local and particular. But when the market falls, it falls for all of them at once, because they read the same record. The same drawdown reaches every awakened Relic in the same moment; what differs is how each wears it. DNA decides that. The market is the civilization's climate; the DNA decides how each body survives the weather.

So the collection changes collectively without becoming uniform. A hard sell-off does not apply one identical filter to ten thousand images. It darkens a fragile Sacred Shard toward collapse while a Buried Monolith merely dims, and it destabilises a Ritual Machine's symmetry in a way a Containment Core resists. A recovery reaches all of them and lifts each differently. Over a long enough history the civilization accumulates something like archaeology in real time: eras of drought and expansion legible across the whole population, worn one way here and another way there, with each owner's permanent inscriptions layered on top.

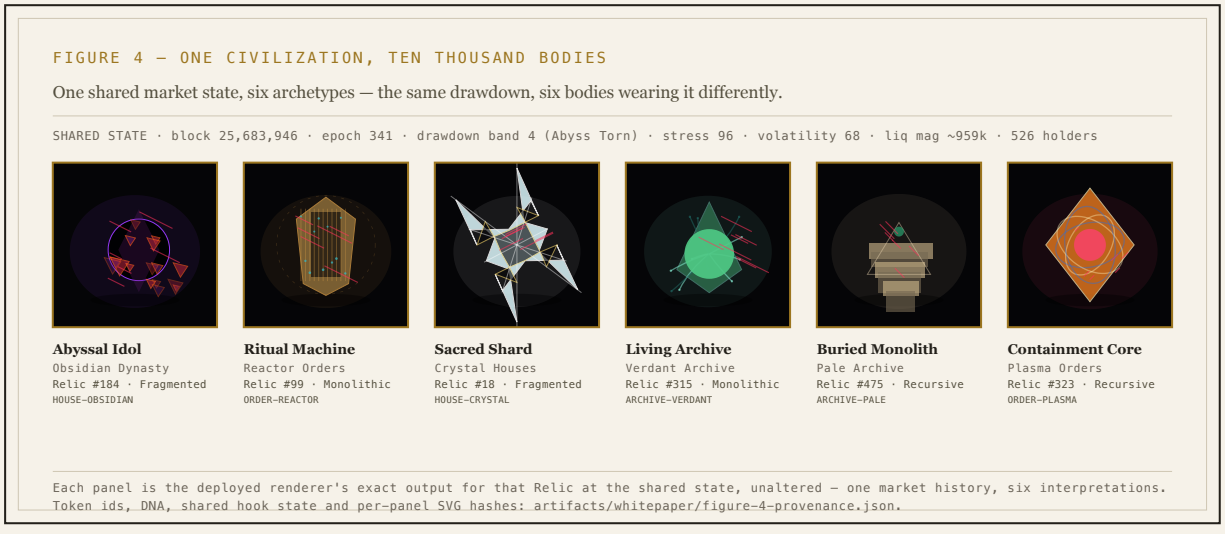


Figure 4. One shared market condition, six archetypes. Six real Relics — one drawn from each archetype, each also crossing a different palette house — rendered by the deployed on-chain renderer under a single hook state recorded on Ethereum mainnet. The same drawdown reaches every body at once; DNA decides how each wears it. No panel was recolored or edited; each is the renderer's exact output for that Relic at that state. Token ids, DNA, the shared state, and per-panel hashes are in the figure's provenance record.

7. Market history as material

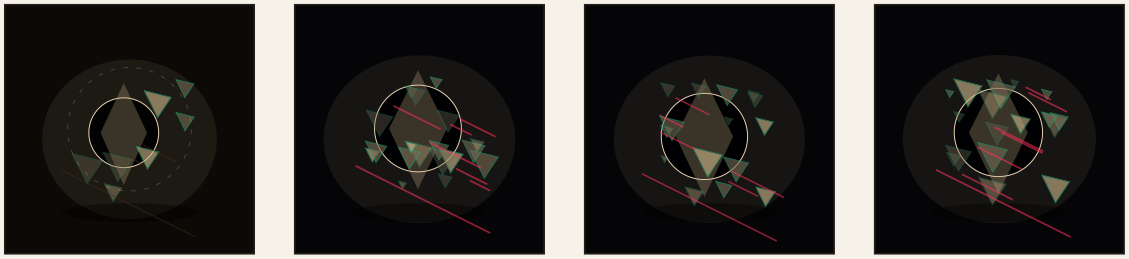
The renderer does not draw a chart. It translates recorded market state into visual and material terms, and the difference matters: a chart shows you the market; a Relic shows you what the market did to an object left standing in it. The mapping, stated only where it is verified in the deployed system:

MARKET CONDITION	RECORDED STATE	MATERIAL INTERPRETATION
Buying pressure, recovery from a low	Buy volume, net flow, recovery reading	Repair, renewed light, resonance
Selling pressure, decline	Sell volume, drawdown beneath the high	Fracture, corruption, darkening
Instability	Volatility, stress	Broken symmetry, unsettled form
Depth in the pool	Liquidity magnitude	Structural density and ground
A new all-time high	All-time-high tick	Expansion; a permanently recorded high-water mark
Distance below the former high	Drawdown band	Depth of the wound; how dark the ground goes
Growth in the number of holders	Active-holder count	Civic expansion across the civilization
An owner's explicit inscription	Scars and resonance, deepened by `evolve`	A permanent local mark, chosen and kept

Two of these rows are permanent and the rest are weather, and the two are never blurred. The all-time-high tick and the owner's inscriptions are laid down and kept. The drawdown, volatility and stress are the current condition; they deepen and they lift. So the market is not only damage. It inflicts weather that passes and it lays down record that stays, and a holder can add permanent record of their own. Pressure becomes fracture; recovery becomes light; a long history becomes sediment.

FIGURE 3 — MARKET HISTORY AS MATERIAL

One Relic across market history — Relic #109, an Abyssal Idol of the Pale Archive, read against four real hook states.



1 Quiet genesis
block 25,678,483
epoch 0 · draw 0
stress 0 · vol 0
liq ~0k · holders 1
Contained · Mythic

2 Volatile, thin liquidity
block 25,679,848
epoch 132 · draw 4
stress 100 · vol 335
liq ~47k · holders 253
Abyss Torn · Genesis

3 Drawdown eases
block 25,681,214
epoch 183 · draw 3
stress 85 · vol 58
liq ~100k · holders 396
Cracked · Genesis

4 Deep-liquidity present
block 25,683,946
epoch 341 · draw 4
stress 96 · vol 68
liq ~959k · holders 526
Abyss Torn · Genesis

Real historical archive reads of the deployed renderer (0x70ecD6…7b8D). DNA and token history are held constant; only market state differs.
Panels are the renderer's exact SVG output, unaltered. Hook state and SVG hashes per block: artifacts/whitepaper/figure-3-provenance.json.

Figure 3. The same immutable Relic — #109, an Abyssal Idol — read against four market states that actually occurred on Ethereum mainnet: a quiet genesis before trading, volatile early trading on thin liquidity, the drawdown easing as liquidity accumulates, and the deep-liquidity present. Its DNA and token history are held constant, so only the recorded market state differs, read live from the deployed renderer at each historical block. As the drawdown deepens the object fractures and scars; as liquidity accumulates its ground gains density. These are real historical archive reads, not simulations — per-block state and hashes are in the figure's provenance record.

8. Participation without a single author

Authorship here is real but distributed, and it is worth being precise about who did what.

The artist authored the rules: the archetypes, the visual grammar, the six identity attributes, the translations from market state to material, and the constraints the whole system runs under. Those decisions were made once and then made permanent. The contracts execute them and can execute nothing else; there is no seat left from which the artist can repaint the work.

Everyone who touches the market makes the history the rules interpret. Buying and selling create the pressure. Providing liquidity sets the structural ground. Holders turn capacity into bodies by awakening Relics, and deepen individual objects by evolving them. Time is not a cause (nothing moves because an hour passed), but it is the axis along which all of this is allowed to happen. Ethereum keeps the rules running.

No participant composes an image. No single trade can claim a Relic's appearance as its work, because the appearance is a function of the whole recorded history, not of any one event within it. This is participation without a vote, without a curator, and without a mutable artist hand: the crowd acts collectively and without a shared intention, and the system interprets the result. Traders do not own the artistic process. They contribute to the history the process reads.

9. The EVM as material

It is tempting to treat Ethereum as mere hosting — the place the artwork happens to live. For RELICS the machine is closer to a material, with the resistances a material has.

The Ethereum Virtual Machine imposes hard limits: a ceiling on how large a contract's deployed code may be, a cost on every operation, deterministic execution with no source of true randomness, finite and expensive storage, bounded work inside each callback, integer arithmetic without floating point, and deployment that cannot be revised. Every one of these shaped the work. Stone limits what a sculptor can carve and pigment limits what a painter can mix; bytecode limits what an on-chain artwork can be, and the discipline shows in the result — the compression, the recurring forms, the reliance on procedural rules rather than stored detail.

The clearest instance is the renderer, which carries the entire drawing system in a single contract and sits 33 bytes under the machine's hard ceiling on deployed code. There is no room to grow into and no upgrade path to grow through. That is a real constraint, not a boast, and it means the visual system is effectively fixed at deployment. The detailed measurements belong in the technical record (Appendix C); what belongs here is the point that the limit is not something the work apologises for. It is the material the work is made of.

10. Immutability without stasis

It helps to say exactly what cannot change and exactly what can, because the value of the work depends on both being true at once.

Fixed. The deployed code, with no proxy and no upgrade path. The supply: exactly ten thousand whole units of $RELICS, minted once, with no function to mint or burn more. The renderer's address, held immutably inside the Relics contract, so the drawing system can never be repointed. Each Relic's DNA. The hook's binding to the one canonical pool, made once and unrepeatable. The hook's three permissions. And the authority state: ownership of the token and the hook has been renounced to the zero address, and the Relics contract and the renderer never had an owner at all. There is no admin key, no pause, no allowlist, no tax, no fee switch, and no owner-controlled metadata anywhere in the system.

Open. The market state the hook keeps writing. The current rendered image, which follows that state. The permanent inscriptions an owner may add through evolution. Ownership of the Relics themselves, as they trade. And each Relic's active or dormant status, as holders awaken and retire them.

Figure 5. The fixed rules — code, supply, renderer, DNA, pool binding, permissions, ownership — form the frame; the mutable state — market record, rendered image, evolution, ownership, dormancy — moves within it.

One more distinction matters for anyone verifying the work. The website at relics.wtf and the listing on any marketplace are *views* — convenient, replaceable, occasionally stale. The contracts are the work's canonical procedure. When a view and the chain disagree, the chain is right. The art does not live on the website. It is computed on Ethereum, and the website only shows you a picture of it.

---

11. Limits of the work

An honest account of what the work does requires an honest account of what it does not.

Market state can be moved by anyone with enough capital; a determined buyer or seller can push the record in a direction. RELICS accepts this because it never uses market state as a financial oracle and never lets it gate a financial outcome — what you can buy, sell, or withdraw does not depend on it, so moving it buys you influence over appearance and nothing else. During a stretch when nothing relevant happens in the pool, the work produces no new visual state: ask a Relic twice and it answers the same, and that stillness is the work holding its place, not the work being finished. Marketplace images can lag the chain, and the website is not canonical. On-chain rendering is computationally bounded, and because the code is immutable it cannot be patched — the constraints of Section 9 are permanent, for better and worse. The genesis liquidity that backs the market is held in a third-party timed lock, described neutrally in Appendix E; what any holder owns — their $RELICS balance and their awakened Relics — is independent of that arrangement. The engineering detail, including custody parameters and every measurement, lives in the technical reference (`TECHNICAL.md`), which is where an integrator or auditor should go next.

12. The work remains open

RELICS is not a static image attached to a token, a token with decorative art, a chart rendered as a picture, or an animation an artist drives. It is an immutable artistic system, bound to one real public market, that keeps that market's history as visual form and stays open to futures no artist or collector can predict.

The identity of each Relic is settled. The rules that draw it are settled. What is not settled is the image, because the market keeps supplying the one input the rules were built to read, and no one (not the artist, not the holder, not the crowd) can decide in advance what that input will be.

Ethereum keeps the rules. The market brings the history. What remains is the Relic.

Appendix A — System topology

COMPONENT	STANDARD	ROLE
$RELICS token	ERC-20	Fixed 10,000-unit supply; the backing resource and the active-holder count
Relics	ERC-721	The artifacts: awakening, dormancy, evolution
Renderer	view contract	Computes JSON + SVG from DNA, token history and market state
Hook	Uniswap v4 `BaseHook`	Records market history into one shared structure
PoolManager	Uniswap v4 singleton	Calls the hook at initialize / add-liquidity / swap
Canonical pool	$RELICS/WETH, fee 0.30%, spacing 60	The one market the hook reads
Genesis position	Uniswap v4 PositionManager NFT	The single liquidity position, in a third-party timed lock

The five contract addresses, the pool id, the position id and the custody details are in Appendix E. Figure 6 shows the same topology as a diagram.

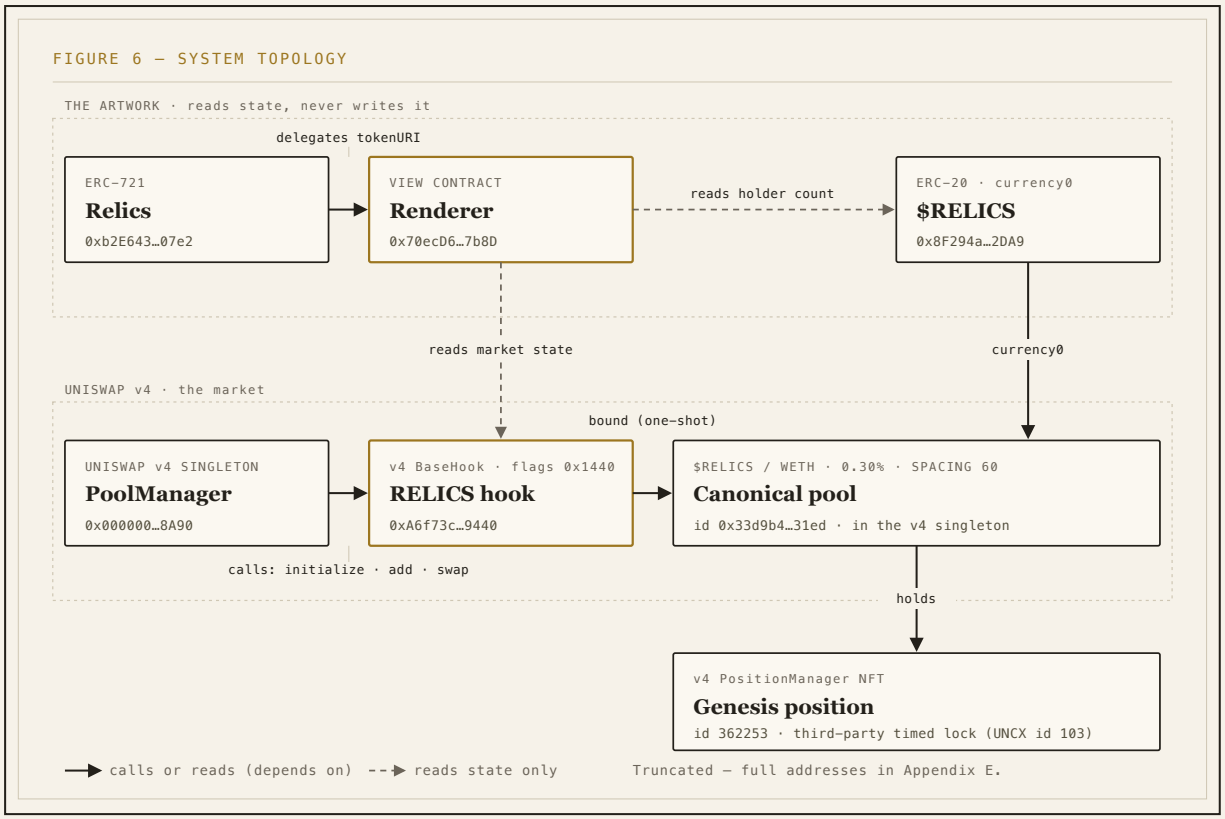


Figure 6. The three deployed contracts, the renderer, and the one canonical pool that binds the market to the artwork. Rendered from the mainnet addresses in Appendix E.

Appendix B — How acquisition and awakening work

This is user truth, not the artistic thesis, and it is kept brief here on purpose.

$RELICS is a fixed supply of ten thousand whole units. Each whole unit you hold establishes capacity for one active Relic; a fractional balance establishes none. Receiving $RELICS — including as the output of a swap — does no work on the artifact layer at all. It raises your latent capacity, which is simply the gap between the room you hold and the Relics you have awakened, and which is never stored.

Awakening is a separate transaction you send yourself. It has no recipient parameter, so no marketplace operator and no token spender can awaken a Relic into your wallet; the Relic is always delivered to the caller. Up to eight Relics may awaken per call. You do not choose which identities you receive and neither does the contract: dormant ids are held as a set and always drawn lowest-first, then fresh ids in order. Because that draw is public you restate it — you name the exact ids your next awakening would produce and the minimum you will accept — and if someone else's transaction lands first your call can be shortened or refused, but it can never hand you an identity you did not name. A refused call writes nothing.

Selling beneath your backing is the mirror. Sending $RELICS away is the only thing that reduces capacity, so if you sell below what you hold, the excess Relics go dormant in the same transaction, most-recently-awakened first, up to a fixed budget; past that budget the transfer reverts and asks you to prepare the sale by choosing which Relics to retire. A dormant Relic is genuinely burned as an ERC-721, but it keeps its DNA, its fingerprint, its original minter and every mark it accumulated, and it returns unchanged when its id is drawn again.

Marketplace transfer carries backing with it: moving a Relic directly moves exactly one whole $RELICS unit alongside it, so neither side is left off its backing line, and an awakened Relic is an ordinary ERC-721 that lists on any marketplace with no wrapper step. The full mechanics — capacity model, the named-id draw, the send budget, the dormancy seal — are specified in `TECHNICAL.md`.

Appendix C — Technical state model

Hook callbacks (three, all observational): `afterInitialize` validates and fixes the opening price and seeds the record; `afterAddLiquidity` observes additions; `afterSwap` records directional pressure and price. No before-hooks, no `afterRemoveLiquidity`, and no return-delta permissions — the hook cannot alter, tax, or refuse a trade.

Shared market state is one bounded structure: swap count, liquidity-event count, last swap block, epoch, cumulative buy and sell volume, cumulative liquidity magnitude, net flow, last tick, all-time-high tick, drawdown band, stress, volatility, and a rolling market seed. It holds no per-token data. Holder count is read separately from the token contract.

Immutable DNA fields (packed in one 256-bit word): species, biome, palette, geometry, energy, temperament, mutation potential, rarity seed, birth epoch, and entropy.

Persistent token history accumulated per Relic through owner evolution: scars, resonance, lineage and evolve count, each monotonic.

Phenotype: thirteen attributes — six fixed from DNA (Species, Archetype, Geometry Class, Glyph Family, Civilization, Energy Signature) and seven recomputed on read (Corruption State, Historical Tier, Liquidity Exposure, Volatility Exposure, Scar Severity, Recovery State, Resonance). Of the seven, only Resonance is strictly monotonic (owner-driven, through `evolve`). Historical Tier is built mostly from permanent inputs (rarity class, age, and accumulated scars, resonance and transfers) but also folds in the current active-holder tier and a reversible deep-drawdown bonus, so it can move down as well as up. Scar Severity mixes a permanent baseline of accumulated scars with current drawdown; Liquidity Exposure reads a cumulative magnitude that only grows; and Corruption State, Volatility Exposure and Recovery State track live, reversible market state.

Limitations: the renderer is fixed at deployment with no upgrade path; a quiet pool produces no new visual state; identical state yields an identical image; market state is art input only and gates no financial outcome. Depth and measurements are in `TECHNICAL.md`.

Appendix D — Immutability matrix

COMPONENT	FIXED	CHANGES OVER TIME	AUTHORITY REMAINING
Deployed code	Yes — no proxy, no upgrade	–	None
$RELICS supply	10,000 units, minted once	–	None (no mint, no burn)
Renderer address	Immutable inside the NFT	–	None
Relic DNA	Fixed at forge	–	None
Hook ↔ pool binding	One-shot, spent	–	None (no unbind/rebind)
Hook permissions	Flags fixed at deploy	–	None
Ownership (token, hook)	Renounced to `address(0)`	–	None
Ownership (NFT, renderer)	Never had an owner	–	None
Market state	–	Written by the hook on relevant events	The market, collectively
Rendered image	–	Follows market state and token history	None (deterministic)
Per-Relic evolution	–	Deepens only, owner-invoked	The Relic's owner
Relic ownership / dormancy	–	Trades, awakens, retires	The holder

Appendix E — Mainnet record

Chain id 1. Launched 3 August 2026. Every row is a public read against Ethereum.

$RELICS token (ERC-20)	`0x8F294a99a0609822C233b24867F331c292cE2DA9`
Relics (ERC-721)	`0xb2E643345013E4944E45777E12fC0EC67A9c07e2`
Renderer	`0x70ecD6Cab03c3fef7F7CCE4ff669c0D0e6e17b8D`
Uniswap v4 hook	`0xA6f73cc88723f04b85E2c2aF3e35F759Dc1A9440`
PoolManager (v4 singleton)	`0x000000000004444c5dc75cB358380D2e3dE08A90`
Canonical pool id	`0x33d9b4089069272e5aeaeccf24bc710a7ee8cf65f4ecde682187a2fc355531ed`
Pair / fee / tick spacing	$RELICS / WETH, 3000 (0.30%), 60
Hook permission flags	`0x1440` — `afterInitialize`, `afterAddLiquidity`, `afterSwap`
Initialized price	0.00024911768873 WETH per $RELICS, at tick −82980
Genesis position	id 362253, WETH contributed 0 (single-sided, asserted zero)
Genesis liquidity	157834625076480800773 at mint; 156256278825715992766 currently held in the lock
Custody	third-party liquidity lock (identified as UNCX lock id 103), lock owner `0x147AECa171a79466Fe9E2c03f21b45155Ff403F8`
Token ownership	renounced — tx `0x01d1abd63c9a51594f1854d861748603ca334b03964a226dfab99db67a7a3a0c`
Hook ownership	renounced — tx `0x7c8540ff700afd9c22276bef915cb732aea78db394fd78285f7b624a9af8fa75`
Relics & renderer ownership	never existed
$RELICS total supply	100000000000000000000, of which the deployer holds none
Renderer runtime size	24,543 bytes (33 under the 24,576-byte ceiling)

Read the current liquidity live with `PositionManager.getPositionLiquidity(362253)`; read market state with the hook's `getGlobalState()`.

Appendix F — Limits, custody, and further reading

The candid limits of the work are stated in Section 11 and not repeated here. On custody: the genesis liquidity position is held in a third-party liquidity lock; a holder's own $RELICS and Relics are independent of it, and the live principal is readable on chain (Appendix E). This paper is the artistic argument; the engineering specification, the exact custody parameters, and every measurement are in `TECHNICAL.md`, which is the correct reference for integrators and auditors. Nothing here is financial advice, an offer, or a promise of value, liquidity, or exit.

Appendix G — References

1. Sol LeWitt, wall drawings as executed instructions. SFMOMA, *Sol LeWitt*, https://www.sfmoma.org/artist/sol_lewitt/ ; Spencer Museum of Art, *Wall Drawing 519*, https://spencerart.ku.edu/art/collections-online/exhibition/1497 . ↩

2. Vera Molnár, early machine and generative practice. Thaddaeus Ropac, "Meet Vera Molnár, the generative art pioneer," https://ropac.net/news/793-meet-vera-molnar-the-98-year-old-generative-art-pioneer/ . ↩

3. Casey Reas, on software and process as the artwork. Reas, "Process," https://reas.com/process/ ; Whitney Museum, *Casey Reas: {Software} Structures*, https://whitney.org/exhibitions/software-structures . ↩

4. On the fragility of software- and computer-based art and the work of preserving it. Guggenheim / NYU, Conserving Computer-Based Art (CCBA), https://www.nyu.edu/about/news-publications/news/2019/february/how-do-we-conserve-and-restore-computer-based-art-in-a-changing-.html . ↩

5. Larva Labs, *Autoglyphs* — the generator embedded on chain; each glyph fixed at mint. https://www.larvalabs.com/autoglyphs . ↩

6. Art Blocks, on fully on-chain generators and on parameters that remain writable on chain after mint (PostParams). https://docs.artblocks.io/protocol/on-chain-storage/ ; https://www.artblocks.io/articles/postparams-debuts-jiwa-opensea-living-art . ↩