S

SwagBall

A programmable funding-cycle protocol with generative on-chain artifacts

WHITE PAPER / DRAFT v0.2

September 2026

Scope

This document describes the current SwagBall concept and the implemented Avalanche FundingManager system, including funding economics, participation and winner NFTs, generative artwork capture, oracle processing, storage, user experience, security assumptions, migration, and the planned multi-network product architecture.

> **Status: testnet software, not a mainnet promise**
> The current reference deployment is on Avalanche Fuji. Winner selection is off-chain and one-entry-per-participating-address; Sybil resistance, production randomness, independent smart-contract audit, legal/regulatory analysis, and production administration controls remain explicit pre-mainnet work items.

swagball.io → avax.swagball.io → fuji.avax.swagball.io

Executive summary

SwagBall is a funding-cycle system designed to turn project funding into a transparent, progressive, and collectible on-chain experience. Participants contribute native network currency into structured cycles. Each cycle has a target, accepts up to 120% of that target, routes 20% of the target to the project, and reserves 100% of the target behind a WinnerNFT claim right. The cycle also produces a generative visual artifact captured from a live canvas at the moment of completion.

The protocol separates deterministic on-chain accounting from intentionally replaceable off-chain services. Funding limits, reserves, project allocations, claims, cycle schedules, participation records, and NFT ownership are enforced by Solidity contracts. Artwork capture, winner selection, and metadata/storage orchestration are performed by an oracle. This separation allows the experience and storage network to evolve without rewriting the core funding logic.

The current Avalanche implementation is deployed to Fuji testnet. Its user experience is intentionally dual-layered: a Standard UI explains and guides participation for first-time users, while a Degen UI exposes richer telemetry and controls for advanced users. The long-term product model separates generic SwagBall education from network-specific applications so the same concept can expand to multiple networks without duplicating the entire experience.

> **Core idea**
> A completed funding cycle creates three linked outcomes: project funding, a reserved winner claim, and a permanent generative artifact associated with that cycle.

Contents

1. Purpose and design principles

Traditional funding pages often collapse the experience into a progress bar and a payment button. SwagBall instead treats funding as a sequence of auditable cycles. Each cycle has explicit economics, explicit participation limits, a visible completion condition, and a resulting artifact. The goal is not merely to process payments, but to make project funding legible as an evolving public system.

- Transparent accounting: cycle targets, caps, raised amounts, reserves, claims, winners, and participant counts are readable on-chain.
- Bounded participation: each wallet is capped per cycle to reduce single-wallet concentration.
- Progressive schedules: funding targets can advance through predefined growth sets or owner-staged custom schedules.
- Cycle identity: each completed cycle can be referenced by a stable cycle ID that also becomes the WinnerNFT token ID.
- Artifact permanence: the live generative image is captured before winner selection so the artwork belongs to the cycle, not to the chosen wallet.
- Replaceable infrastructure: storage and randomness sources can evolve behind narrow interfaces without rewriting the core cycle accounting.
- Two-layer UX: simple participation for newcomers and deeper telemetry for advanced users.

2. System overview

The current Avalanche implementation consists of three contracts plus an off-chain oracle, storage adapter, visualizer, and web application.

Component	Responsibility
FundingManager	Accepts AVAX, enforces cycle and wallet limits, completes cycles, routes the project share, tracks reserves, records participation, advances schedules, and pays WinnerNFT claims.
WinnerNFT	ERC-721 minted one-per-completed-cycle to the selected participant. Token ID equals cycle ID and ownership controls the reserved-fund claim.
VoucherNFT	ERC-721 minted on a wallet's first contribution to a cycle. It can optionally be configured as a burnable voucher for a governance token.
Oracle	Watches CycleCompleted events, captures the current visualizer image, loads participants, selects a winner, writes metadata, and fulfills the cycle on-chain.
Visualizer/UI server	Produces and streams the live 480x320 generative canvas and exposes a snapshot endpoint used by the oracle.
Storage adapter	Persists image and metadata bytes and returns stable public URIs. Local HTTP-backed storage is used on Fuji; an IPFS-like adapter is planned.
Standard UI / Degen UI	Two presentations of the same system: guided newcomer experience and advanced operational/visual experience.

CONTRIBUTE → CYCLE FILLS → ARTWORK CAPTURE → WINNER SELECTED → WINNER NFT / CLAIM

3. Funding-cycle economics

Each cycle snapshots its economics when the cycle starts. Later schedule changes do not alter historical liabilities. For a cycle target T, the deployed contract uses the following fixed ratios:

Quantity	Formula	Meaning
Cycle target	T	Amount reserved for the eventual WinnerNFT holder.
Cycle maximum	1.20 × T	The cycle completes only when total accepted contributions reach 120% of target.
Winner reserve	1.00 × T	Retained by FundingManager as the cycle's claim liability.
Project share	0.20 × T	Paid to the project wallet when the cycle completes, or accrued if the payment fails.
Per-wallet cap	0.20 × T	Maximum accepted from one wallet within the cycle.

Because one wallet can contribute at most 20% of target while the cycle must reach 120%, at least six fully capped addresses are required to complete a cycle. In practice, more participants may be required if individual contributions are smaller than the cap.

> **Important distinction**
> Contribution size affects how much AVAX a wallet puts into the cycle, but the current Fuji winner-selection model gives one winner entry per participating address, not one entry per AVAX. This creates a known Sybil/address-splitting risk and is explicitly not considered final mainnet economics.

Contribution behavior

A contribution is accepted only up to both the active cycle's remaining capacity and the sender's remaining per-wallet cap. Excess value is refunded. If a contribution fills a cycle and still has value remaining, the contract can continue processing against the newly opened next cycle, subject to that cycle's limits. The Standard UI intentionally constrains normal submissions to the reviewed cycle to avoid accidental spillover; the contract itself remains the final authority.

Project payment behavior

At cycle completion, the contract advances to the next cycle before attempting the project payment. The project transfer uses a bounded gas allowance and is deliberately non-blocking: if the project wallet rejects the transfer, the amount is recorded as projectAccruedFunds and can later be withdrawn by the project wallet or owner. A rejecting project smart wallet therefore cannot prevent cycle completion.

4. Cycle schedules and growth sets

Targets are organized into phases. A phase specifies how many cycles run at a given target. The project includes five explicit built-in growth sets. When automatic set advancement is enabled, completing the final phase of a built-in set stages the next set; Set 5 is terminal and repeats. Owners can also stage custom schedules, but staged changes activate only at cycle boundaries.

> **Snapshot rule**
> The current cycle never changes economics mid-cycle. Target, 120% maximum, and wallet cap are snapshotted when the cycle opens.

5. Participation vouchers

On the first accepted contribution by an address in a cycle, FundingManager records that address as a participant and mints one VoucherNFT to it. Additional contributions by the same address in the same cycle do not create additional participant entries or additional vouchers.

VoucherNFT is an ERC-721 with a configurable governance-token redemption mechanism. The contract records the actual minted token ID, cycle ID, and original participant. The public metadata service uses those immutable identifiers to create a deterministic participation artifact. The contract owner can also configure an ERC-20 governance token and a voucher value; a holder may then burn the NFT to receive the configured amount, assuming sufficient token inventory.

> **Fuji status**
> The current Fuji deployment has the governance token configured as the zero address, so voucher-for-governance redemption is not active. The redemption path is a protocol capability, not a current Fuji promise.

Because VoucherNFT uses standard ERC-721 behavior, vouchers are transferable unless a future version introduces additional restrictions. The original participant and cycle identity remain recorded for the token even if ownership changes. Current Fuji metadata resolves to a lightweight deterministic PNG and a token-specific SwagBall artifact page.

6. WinnerNFT and claim mechanics

Each completed cycle reserves exactly the cycle target T. After the oracle selects a winner, WinnerNFT mints token ID equal to the cycle ID directly to that participant. The NFT is therefore both a collectible artifact and an on-chain claim instrument tied to the reserved cycle funds.

Partial claim

The current WinnerNFT holder may claim up to 50% of the target without surrendering the NFT. Partial claims update claimedAmount and reduce the global reserved-fund accounting by the exact amount paid.

Full claim

To receive all remaining cycle funds, the WinnerNFT holder approves FundingManager to transfer the NFT. FundingManager pays the unclaimed balance and transfers the WinnerNFT to the project wallet. If a 50% partial claim occurred earlier, the full claim pays only the remaining 50%.

> **Claim rights follow NFT ownership**
> The contract checks the current owner of WinnerNFT at claim time. Because the NFT is transferable, the remaining claim right can move with the token, subject to any amount already claimed for that cycle.

7. Generative artwork lifecycle

The live visualizer is part of the protocol experience but not part of Solidity consensus. It continuously produces a 480x320 generative canvas that participants can see in both the Standard and Degen interfaces. When a cycle completes, the oracle captures the current visual state before selecting the winner.

1. FundingManager emits CycleCompleted(cycleId).
2. The oracle requests the current image bytes from VISUALIZER_SNAPSHOT_URL.
3. The image is persisted through the configured storage adapter.
4. Only after image storage does the oracle load the participant list and select the winner.
5. Metadata is created with cycle ID, winner, target, image hash, and capture timestamp.
6. The oracle calls fulfillRandomWinner(cycleId, winner, tokenURI).
7. WinnerNFT token ID = cycle ID is minted to the winner.

Capturing first is a deliberate sequencing decision: the visual artifact represents the completed funding cycle and cannot be influenced by knowing which wallet won.

8. Oracle and winner selection

Winner selection is currently orchestrated by an off-chain Node.js oracle authorized by FundingManager. The oracle waits for configurable chain confirmations, polls CycleCompleted events in bounded block ranges, persists its progress, and is designed to retry safely if fulfillment is interrupted.

Participant set

FundingManager stores the unique participating addresses for each cycle. The oracle pages through that on-chain list and chooses one index. Each address appears once regardless of contribution amount.

Random source

The oracle supports two configured random sources: Node.js cryptographic randomness (crypto.randomInt) and RANDOM.ORG. When RANDOM_SOURCE=random.org, the current implementation uses RANDOM.ORG signed integers, persists the signed random object and signature before fulfillment, and embeds that drawing provenance in WinnerNFT metadata. The public Transparency view can submit the stored proof to RANDOM.ORG verifySignature. This improves auditability and non-repudiation, but winner fulfillment still depends on an authorized off-chain oracle and is not a trust-minimized on-chain randomness scheme.

> **Mainnet decision required**
> Production randomness is an explicit unresolved design decision. The current oracle operator can influence or withhold fulfillment and the one-address-one-entry model can be Sybil-attacked. A production launch requires a documented randomness and participant-identity/weighting decision.

9. Storage and metadata architecture

SwagBall separates image generation from image storage. The visualizer is the source of artwork bytes; a storage adapter decides where those bytes and the matching metadata are persisted.

Layer	Current Fuji implementation	Intended evolution

Image source	Live UI visualizer snapshot endpoint	Same conceptual source; renderer may evolve.
Storage backend	Local filesystem served at /nft/... over HTTPS	IPFS-like/content-addressed storage adapter.
Token URI	Stable HTTP metadata URL on fuji.avax.swagball.io	Production host on avax.swagball.io; optionally content-addressed URI if migration/freeze semantics are added.
Integrity	SHA-256 of image recorded in metadata/manifests	Content addressing and/or decentralized replication.

The storage interface is intentionally narrow: putImage(...) and putMetadata(...), each returning a stable URI and SHA-256 digest. This allows a future decentralized storage network to replace local persistence without changing oracle orchestration.

10. User experience: Standard and Degen

The Avalanche application uses two interfaces over the same underlying system rather than maintaining separate protocol implementations.

Standard UI	Degen UI
Default first-time experience.	Advanced/experimental interface preserved at /degen.
Explains the current cycle, 100%/120% economics, wallet cap, transaction review, activity, artifacts, and transparency.	Exposes richer telemetry, visualizer controls, music, funding controls, and lower-level state.
Uses the live NFT generator canvas as a central product object.	Uses the same generative stream as an expressive operational surface.
Treats wallet/network/RPC errors as explicit user states.	Optimized for users who already understand the system.

This separation is a product principle: contract complexity should power the experience without requiring a newcomer to understand contract internals before participating.

11. Multi-network product architecture

The planned product architecture separates generic SwagBall understanding from network-specific execution. This avoids cloning the same explanatory content for every chain.

Domain	Role
swagball.io	Network-agnostic SwagBall story: what the system is, how cycles and artifacts work conceptually, supported networks, and ecosystem activity.
avax.swagball.io	Avalanche production application: AVAX-specific contracts, live funding, artifacts, activity, and Avalanche transparency.
fuji.avax.swagball.io	Avalanche Fuji testnet environment using the same application model with testnet configuration and prominent testnet signaling.
.../degen	Advanced UI route for the relevant network/environment.

Future network applications should share core concepts and interface components while keeping chain-specific configuration—native asset, chain ID, RPC endpoints, contracts, explorer links, deployment addresses, and storage hostnames—isolated. The current implementation is Avalanche-first; multi-network deployment is a product direction, not yet a claim of chain parity.

12. Administration, migration, and accounting safety

FundingManager uses Ownable, Pausable, and ReentrancyGuard from OpenZeppelin and maintains explicit accounting for winner reserves, accrued project funds, the active cycle, and surplus.

- Oracle and project wallet addresses are owner-configurable.
- Contributions can be paused and unpaused; a migrated manager cannot be unpaused.
- Custom phases or built-in growth sets can be staged, cancelled, and activated only at cycle boundaries.
- Accrued project funds may be withdrawn by the project wallet or owner.
- Surplus recovery is owner-only, requires the contract to be paused, and cannot consume accounted liabilities.
- Direct AVAX transfers to the contract are rejected; contributions must use contribute(...).
- Winner and voucher NFT contract ownership is held by FundingManager so mint authority follows the manager lifecycle.

Migration model

Migration is intentionally constrained. FundingManager can finalize migration only while paused, on a fresh current cycle with zero raised funds, with no unresolved WinnerNFT mint, no pending phases, and enough contract balance to cover historical winner reserves plus accrued project funds. NFT contract ownership then transfers to the replacement manager. Historical reserve liabilities stay in the old manager so existing WinnerNFT holders continue claiming from the contract that holds their funds.

13. Security and trust assumptions

Area	Current model / risk
Smart contracts	OpenZeppelin primitives are used, but the project has not represented the current contracts as independently audited. Independent audit is required before mainnet.
Randomness	Authorized off-chain oracle; RANDOM.ORG signed drawings provide externally verifiable provenance when configured, but oracle availability/withholding and fulfillment authority remain trust assumptions.
Sybil resistance	One entry per address can be gamed by splitting participation across addresses. Explicitly unresolved for mainnet.
Owner/admin	Owner can change oracle/project wallet, schedules, pause state, voucher settings, and perform constrained migration/surplus recovery. Production ownership should be behind an appropriate multisig/governance process.
Oracle key	A privileged private key can call winner fulfillment. It must be isolated, rotated if exposed, and operated with strong monitoring.
Storage availability	Fuji metadata/images rely on web-hosted local storage. Availability and immutability are weaker than content-addressed decentralized storage.
Legal/regulatory	Prize-like winner selection, transferable claim NFTs, and multi-jurisdiction deployment can create legal obligations. Specialist legal analysis is required before production availability.

14. Current Fuji reference deployment

The following values describe the current Fuji deployment created September 9, 2026 at deployment block 58,289,827. They are testnet references, not permanent production identifiers.

Item	Fuji value
Network / chain ID	Avalanche Fuji / 43113
FundingManager	0xDDbc7feE1f3E468fa3C7681b0e9aB4E2D76AfEb3
WinnerNFT	0x2297C28ED34F11Da7205921dbf030D60F2cC47b8
VoucherNFT	0x8BA635C072BC3a98b16d5Bc2B86B7efe9D335fd9
Project wallet	0x51d65785193d1DD7F4AbF1Ca3fB109AcA434B7E3
Oracle	0xa975c5c87389Ebc7aAA5b7AA5767E71181e535fd
Initial built-in growth set	Set 1 - Launch
Governance token	Zero address (voucher redemption not configured)
Public environment	fuji.avax.swagball.io
Deployment block	58,289,827

15. Roadmap to production

1. Continue Fuji user testing with people who have not previously seen the project; refine comprehension, failure states, and transaction confidence.
2. Complete independent Solidity security audit and adversarial economic review.
3. Choose and implement production winner randomness with a documented trust model.
4. Resolve Sybil resistance / participant weighting policy and test it against economic attacks.
5. Move owner administration to an appropriate multisig and document operational controls.
6. Complete legal/regulatory analysis for winner selection, claim NFTs, voucher behavior, and supported jurisdictions.
7. Decide production NFT-storage guarantees: stable HTTP gateway, content-addressed network, URI freeze/migration semantics, or a combination.
8. Deploy and validate avax.swagball.io with production contract/config isolation from Fuji.
9. Launch swagball.io as the network-agnostic education and ecosystem layer.
10. Generalize chain configuration and deployment tooling before adding additional networks; avoid duplicating protocol/business logic per chain.

16. Risks and limitations

SwagBall is experimental software. The current testnet design deliberately exposes unresolved issues rather than hiding them behind product polish.

- Participation does not guarantee selection, payout, appreciation, or any financial return.
- Current address-based winner entries are vulnerable to Sybil behavior.
- Current randomness depends on an authorized off-chain oracle.
- The visualizer and HTTP storage path are operational dependencies outside the EVM.

- Administrative powers remain meaningful and require stronger production controls.
- Native-asset funding and winner/prize mechanics may be regulated differently across jurisdictions.
- Smart-contract bugs, RPC outages, wallet errors, oracle outages, storage outages, and key compromise can impair operation.
- Future schedule changes can alter new-cycle targets, though active and historical cycle economics are snapshotted and protected from retroactive schedule changes.
- Multi-network expansion introduces bridge-free but duplicated-liquidity/community fragmentation concerns unless the product clearly defines how network instances relate to one another.

> **No investment representation**
> This white paper is a technical and product description of an experimental system. It is not an offer of securities, a promise of profit, financial advice, or a guarantee of production launch.

17. Conclusion

SwagBall combines a simple funding primitive with a deliberately visible lifecycle: people fund a bounded cycle; the cycle reaches a deterministic completion threshold; the project receives a predefined share; the winner reserve remains accounted on-chain; a generative image is frozen as the cycle artifact; and a WinnerNFT carries the remaining claim right.

The project's strongest architectural property is separation of concerns. Solidity protects cycle accounting and claims. The oracle coordinates replaceable off-chain services. Storage can migrate behind a narrow adapter. The visualizer can evolve without changing historical cycle economics. Standard and Degen interfaces can present the same protocol to different audiences. The planned domain hierarchy can then extend that model to additional networks without duplicating the project's core explanation.

The next phase is not to hide the experimental parts, but to harden them: audit the contracts, finalize randomness and Sybil policy, strengthen administration and storage, validate the user experience on Fuji, and only then promote the system to production networks.

Appendix A. Contract surface and key events

Surface	Purpose
contribute(voucherURI)	Accept native AVAX subject to active-cycle and per-wallet limits; mint first-contribution voucher; refund excess.
getCurrentCycleState()	Return active cycle ID, phase position, target, maximum, raised amount, wallet cap, growth set, and schedule type.
fulfillRandomWinner(...)	Oracle-only winner finalization and WinnerNFT mint.
claimPartial(cycleId)	Current WinnerNFT holder claims remaining amount up to 50% of target.
claimFunds(cycleId)	Current WinnerNFT holder claims all remaining reserve and transfers NFT to project wallet.
schedulePhases / scheduleGrowthSet	Stage schedule changes for a future cycle boundary.
pauseContributions / unpauseContributions	Administrative circuit breaker.
finalizeMigration(newManager)	Constrained manager migration while preserving historical liabilities in the old manager.
availableSurplus()	Balance not committed to active-cycle funds, winner reserves, or accrued project funds.

Key events

- ContributionReceived / ContributionRefunded
- CycleCompleted
- ProjectPaid / ProjectPaymentAccrued / AccruedProjectFundsWithdrawn
- RandomWinnerSelected
- PartialFundsClaimed / FundsClaimed
- PhasesScheduled / PhasesActivated / PendingPhasesCancelled
- OracleUpdated / ProjectWalletUpdated
- ContributionsPaused / ContributionsUnpaused
- SurplusRecovered / MigrationFinalized

Appendix B. Built-in growth-set schedule

Set	Name	Phase schedule (cycles @ target AVAX)
1	Launch	5 @ 1, 4 @ 2, 3 @ 3, 2 @ 4, 1 @ 5
2	Early	10 @ 5, 8 @ 10, 6 @ 15, 4 @ 20, 1 @ 25
3	Growth	20 @ 10, 15 @ 20, 10 @ 30, 5 @ 40, 1 @ 50
4	Scale	40 @ 15, 30 @ 30, 20 @ 50, 10 @ 75, 1 @ 100
5	Mature	100 @ 25, 50 @ 50, 20 @ 100, 10 @ 500, 1 @ 1000

When automatic built-in-set advancement is enabled, Sets 1 through 4 advance to the next set after the full active set completes. Set 5 repeats. Owner-staged schedules take precedence over automatic advancement at the boundary.

Appendix C. Terminology

Term	Meaning
Cycle	One bounded funding round with snapshotted target, maximum, and wallet cap.
Target (T)	Base cycle amount reserved for the WinnerNFT claim.
Cycle maximum	120% of target; completion threshold.
Participant	Address with at least one accepted contribution in a cycle.
VoucherNFT	First-contribution ERC-721 receipt/voucher for a cycle.
WinnerNFT	Cycle-ID ERC-721 minted to the selected participant; current ownership controls claim rights.
Growth set	Predefined five-phase schedule of cycle counts and targets.
Oracle	Authorized off-chain service that captures artwork, selects a participant, writes metadata, and fulfills the winner.
Artifact	Generative image captured from the live visualizer at cycle completion.
PAQ	Pre-Answered Questions: answers intentionally provided before a project has accumulated frequently asked questions. SwagBall uses PAQ instead of FAQ for first-time education.

Appendix D. PAQ - Pre-Answered Questions

PAQ means Pre-Answered Questions. SwagBall uses this term deliberately: the project is new, so it would be inaccurate to describe these as frequently asked questions. PAQ anticipates the questions a first-time participant, builder, or reviewer should have answered before needing to ask them.

Why does SwagBall use PAQ instead of FAQ?

Because the project does not yet have a history of frequently asked questions. PAQ states plainly that these are questions the project has chosen to answer in advance.

What is a SwagBall funding cycle?

A funding cycle is a bounded on-chain round with a snapshotted target, a 120% completion maximum, and a per-wallet cap. The cycle creates project funding, a winner reserve, participation records, and a visual artifact.

What does a ParticipationVoucher represent?

It is an ERC-721 minted on an address's first accepted contribution to a cycle. Its identity is tied to the actual token ID, cycle ID, and original participant, with deterministic public artwork and metadata.

How is a cycle winner selected on Fuji?

The FundingManager stores one participant entry per participating address. The authorized oracle selects one index. When configured for RANDOM.ORG, the signed drawing object and signature are preserved in WinnerNFT metadata so the drawing can be independently verified.

Does contributing more increase the chance of selection?

No under the current Fuji rules. Each participating address appears once in the selection set regardless of contribution amount. The project treats Sybil resistance and production weighting policy as unresolved mainnet design questions.

What is the difference between ParticipationVoucher and WinnerNFT?

ParticipationVoucher records first participation in a cycle. WinnerNFT is minted to the selected participant after a completed cycle and carries the remaining on-chain claim right for that cycle.

Why are there Standard and Degen interfaces?

Standard UI explains the protocol for a first-time participant and guides funding safely. Degen UI preserves the richer operational and visual interface for users who want lower-level state and controls. Both use the same deployed protocol.

What belongs on swagball.io versus a network subdomain?

swagball.io is the network-agnostic explanation and ecosystem layer. A network site such as avax.swagball.io contains chain-specific funding, contracts, artifacts, and activity. Fuji is the Avalanche testnet environment for validating that experience.

END OF DRAFT v0.2