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, the separate ParticipationVoucher governance design, 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; production address-entry economics, 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. Winner selection is one position per participating address rather than weighted by contribution size. The protocol treats addresses as participation units rather than human identities: multiple-wallet participation is allowed, while the 20% per-address cap turns heavy concentration into a capital-versus-upside tradeoff when those addresses are used to consume cycle capacity. ParticipationVouchers also provide a separate governance path: vouchers may repeatedly second public petitions without being consumed, while binding votes permanently burn vouchers for equal units of weighted voting power. The WinnerNFT is not merely a collectible image: current ownership is the transferable permission key to that cycle's remaining reserve. The live visual engine is an open-ended real-time generator capable of producing up to 75 unique candidate visual states per second; one completion-state frame is committed as each cycle's permanent artifact.

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 address is capped per cycle to reduce concentration by any one address. The cap does not claim that one address equals one person.
- 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. Unspent vouchers are transferable governance weight: they can second petitions without being consumed and can be permanently burned for binding voting weight.
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.
SwagBallGovernance	Independent governance contract for public petitions, cycle-number second quotas, voucher-weighted seconding, voucher-burning ballots, weighted ranked-choice tallying, and deterministic runoffs. FundingManager is only a read-only cycle/project-wallet source.
VoteReceiptNFT / MotionRecordNFT	Non-transferable governance-history NFTs. VoteReceiptNFT records a wallet's consumed voucher weight and ranking; MotionRecordNFT archives a finalized motion to the project wallet. Neither carries governance weight.

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.

> **Funding capacity is not an odds multiplier**
> Contribution size does not buy additional winner entries. Each participating address appears once. More accepted funding can affect shared odds only indirectly by consuming fixed cycle capacity and potentially limiting how many additional addresses enter before completion.

Capacity, participation, and selection odds

The current selection model is not weighted by native-asset volume. A participating address occupies one selection position whether it contributes a very small accepted amount or reaches its full per-wallet cap. Within the final participant set for a completed cycle, a small contributor and a full-cap contributor therefore have the same per-address selection chance.

Contribution size matters in a different way: the cycle has fixed funding capacity. The cycle completes at 120% of target, and one wallet can occupy at most 20% of target. More accepted value from an address that is already participating consumes more of the remaining capacity, leaving less funding opportunity for additional addresses to enter before completion. If fewer addresses ultimately fit into the cycle, the same one-entry-per-address selection is shared across a smaller set. If participants contribute below their caps, more addresses can fit and the odds are spread across a larger set. This is an indirect capacity effect, not additional entries for a larger contributor.

Illustrative examples: if exactly six addresses each use the full 20% cap, the cycle reaches 120% with six participant entries, so each address represents 1/6 of the selection positions (about 16.67%). If twelve addresses collectively fill the same 120% cycle maximum, the final set has twelve entries and each address represents 1/12 (about 8.33%). Individual contribution amounts may differ; each participating address still appears once.

The 20% per-address cap is a concentration limit, not a proof-of-personhood rule. Under the current rules, one address cannot occupy the entire cycle. Six fully capped addresses are the mathematical minimum needed to complete a 120% cycle, while smaller contributions require more participating addresses. SwagBall intentionally counts addresses rather than human identities, so separate participating addresses are separate selection positions even when one controller owns more than one wallet.

Multiple addresses and concentration economics

The protocol does not prohibit multiple-wallet participation. Instead, when one controller uses several addresses to consume more of the same cycle, the controller can increase the number of controlled selection positions only by exposing more aggregate capital to a winner reserve that remains fixed at the cycle target T. The intended mechanism is therefore economic concentration pressure rather than identity policing.

Example - four full-cap addresses controlled by one participant: if Participant A controls four addresses and each contributes the full 20% cap, A contributes 0.80 T and occupies four selection positions. If two independent full-cap addresses contribute the remaining 0.40 T, the cycle closes at 1.20 T with six total positions. Participant A then controls 4/6 of the selection positions. If one of A's addresses wins and the WinnerNFT is fully redeemed with no prior partial claim, the maximum reserve payout is 1.00 T. Relative to A's 0.80 T aggregate contribution, the maximum cycle outcome is +0.20 T before gas or other costs. Each outside full-cap address risked 0.20 T for the possibility of receiving the same 1.00 T target reserve.

Extreme case - six full-cap addresses controlled by one participant: one controller can fill all six 20% positions, contribute the entire 1.20 T cycle maximum, and guarantee that one controlled address is selected because every selection position is controlled. But the WinnerNFT reserve remains 1.00 T. Full redemption therefore returns at most T and leaves a guaranteed 0.20 T net funding cost before gas or other costs. In this full-cap scenario, guaranteeing the win also guarantees the 20% over-target funding loss.

This is the intended tradeoff: more controlled positions can mean more selection probability, but using those positions to occupy more funding capacity reduces reward efficiency because the payout reserve does not scale with the number of addresses a controller owns.

Current Fuji limitation

The current contract creates an address entry on its first accepted contribution and accepts any positive contribution amount. As a result, low-value address splitting can create additional selection positions without consuming the full 20% cap per address. The cap is therefore an economic concentration mechanism, not a

complete Sybil defense. Address-level participation is intentional, but the minimum economic cost of creating additional entries remains a production-design consideration before mainnet.

Cycle duration and participation tempo

The current design has no countdown-based close. A cycle remains open until accepted contributions reach the 120% completion threshold. Depending on participation, a cycle can complete in seconds or remain open for days, weeks, or longer. When the threshold is reached, the cycle completes and one winner is selected from all participating addresses recorded for that cycle.

High participation has two different effects that should not be confused. Within one cycle, a larger final participant set means the one-entry-per-address odds are divided among more addresses. Across time, however, heavier participation can complete cycles faster, creating more completed cycles and therefore more winner-selection events over the same period.

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 and governance

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 that 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 deployed contract also contains a configurable legacy post-burn transfer hook. SwagBall governance uses that existing hook only as burn compatibility: approved vouchers are transferred into the separate

SwagBallGovernance contract and permanently burned for voting weight, while the governance endpoint returns success without issuing a replacement governance token or other value.

> The current Fuji reference deployment predates the separate governance deployment and still records the VoucherNFT governance endpoint as the zero address. The governance deployment script can point the existing burn hook at SwagBallGovernance without changing FundingManager code or redeploying the existing VoucherNFT.

Because VoucherNFT uses standard ERC-721 behavior, unspent vouchers are transferable. The original participant and cycle identity remain recorded for the token even if ownership changes. Current ownership, rather than original ownership, supplies seconding and voting authority. Current Fuji metadata resolves to a lightweight deterministic PNG and a token-specific SwagBall artifact page.

Participation governance

SwagBall governance is intentionally isolated from FundingManager. FundingManager does not execute petitions, seconding, ranked voting, runoff decisions, or governance NFTs. SwagBallGovernance reads the active FundingManager only for the current cycle number and project wallet and verifies that any replacement cycle source references the same VoucherNFT. Funding-cycle accounting, winner reserves, claims, and winner selection remain independent of governance.

Public petitions and voucher-weighted seconding

Any address may create a petition. At creation, the contract snapshots the active funding cycle C and permanently sets that petition's seconding quota to C. One currently owned ParticipationVoucher contributes one seconding unit. A holder with 25 vouchers may therefore contribute 25 seconds. Seconding does not transfer, lock, or burn the voucher, but each specific voucher can second a specific petition only once. The same unspent voucher may second other petitions later.

An unqualified petition receives at least seven days to collect seconds. After that minimum, it expires once the protocol has advanced three funding cycles beyond its creation cycle. A 30-day hard stop applies even if funding cycles stall. Reaching the quota converts the petition into a qualified motion; an expired petition cannot carry its seconds into a later re-submission.

Binding votes: vouchers are exchanged for weight

A wallet may submit one ballot per motion and chooses how many ParticipationVouchers to commit. Each committed voucher is transferred into SwagBallGovernance and permanently burned. One burned ParticipationVoucher equals one unit of ballot weight. No replacement token is minted, and the governance receipt NFTs described below have zero voting weight. This makes seconding a reusable signal while binding voting is a destructive commitment of accumulated participation weight.

Weighted ranked-choice voting

Two-outcome motions use weighted majority voting. Motions with three or more outcomes use weighted instant-runoff voting. A ballot contains an ordered preference list and a weight equal to the number of vouchers burned with that ballot. If the ballot's current preference is eliminated, the same weight moves to its next surviving ranked choice; weight is transferred, never duplicated. Voters may submit a partial ranking. If every ranked option is eliminated, that ballot becomes exhausted and is removed from active voting weight.

Elimination ties are resolved first by comparing the tied options' weight in the immediately preceding round and then walking backward through earlier rounds. If the tie is exact through the available history, the contract enters RunoffRequired rather than using randomness or administrative discretion. A fresh voucher-funded runoff opens only among the tied options. The runoff winner is the tied option that advances; the other tied options are eliminated from the parent round, after which the parent's original ballots continue transferring to

their next surviving preferences. A runoff that itself remains exactly tied can require another runoff. If a runoff ends with no active voting weight, that no-decision result is archived and the parent may open a replacement runoff.

Governance receipt and archive NFTs

Each voting wallet receives one non-transferable VoteReceiptNFT for that motion. It records the motion ID, petition hash, voter, number of ParticipationVouchers burned, and ranked preference order. When a governance vote completes, one non-transferable MotionRecordNFT is minted to the current project wallet to archive the event, including petition hash, total voucher weight, ballot count, result, tally rounds, and runoff linkage when applicable. A vote that ends with no active voting weight is archived as NO DECISION rather than disappearing from the record. Neither NFT can second petitions or cast votes; only unspent ParticipationVouchers are governance weight.

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 selected wallet is the initial owner, but the claim right is attached to the NFT rather than permanently attached to the original winner. The WinnerNFT is therefore both a collectible artifact and a transferable on-chain claim instrument tied to that cycle's reserved funds.

Why the NFT exists: the key-to-the-safe model

SwagBall's product thesis is intentionally blunt: if the NFT stops at the picture, you're doing it wrong. The generated image is public media; it is not the lock. WinnerNFT uses ERC-721 ownership because the protocol needs a unique, transferable permission object that FundingManager can verify, approve, transfer, and redeem.

Conceptually, the NFT is the key and FundingManager is the safe. You can copy the pixels; you cannot copy the permission. At mint, before any partial claim, the token's current owner already controls a protocol-defined claim right that can equal the completed cycle's full target T. The utility exists at mint rather than depending on a future roadmap. This contract-defined redemption right should not be confused with a guaranteed secondary-market price.

Generative throughput and claim value are separate. The live visual engine can produce up to 75 unique candidate visual states per second, but these are not 75 minted NFTs per second. One completion-state frame is captured for a completed cycle and becomes that cycle's WinnerNFT artwork. The claim right comes from cycle accounting and current token ownership, not from image scarcity.

Ownership and live claim access

When a wallet connects to the applicable SwagBall network site, the interface reads on-chain WinnerNFT ownership and claim state. If the connected wallet currently owns an eligible WinnerNFT, the site can expose the live claim actions available for that token. The interface is a convenience layer; eligibility, balances, approvals, transfers, and payments remain enforced by the deployed contracts.

Transfer and holding

WinnerNFT uses standard ERC-721 ownership semantics and may be transferred to another address that can receive the NFT on the supported network. On Avalanche, that means an Avalanche-compatible address. After transfer, the new NFT owner becomes the address entitled to exercise the token's remaining claim right. The original selected winner no longer controls that claim unless the NFT is transferred back.

A holder may simply retain the WinnerNFT without claiming. Economically, the token carries a cycle-specific redemption right backed by the still-reserved, unclaimed portion of the cycle target. This should not be interpreted as a guaranteed secondary-market price: market value is determined independently, while the contract-defined claim right is determined by the cycle target and any amount already claimed.

Partial claim

The current WinnerNFT holder may claim up to 50% of the cycle target without surrendering the NFT. The holder receives that payment and continues to own the WinnerNFT. The contract records the amount already claimed and reduces reserved-fund accounting by the exact amount paid. If the NFT is transferred after a partial claim, the next owner receives the NFT together with only the remaining unclaimed right; previously paid value is not recreated by transfer.

Full claim

To fully redeem or "cash in" the WinnerNFT, the current holder first approves FundingManager to transfer that specific NFT. Through the full-claim transaction, the holder exchanges the WinnerNFT for the token's remaining reserved balance: FundingManager transfers the WinnerNFT to the project wallet and pays the current holder the unclaimed amount for that cycle. If no partial claim occurred, the remaining balance can equal the full cycle target T. If a 50% partial claim occurred earlier, full redemption pays only the remaining 50% of T. After full redemption, the project wallet owns the WinnerNFT and no further claim remains for that token.

> **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.

Claim-right lifecycle example

For a cycle target T = 10 AVAX, the selected winner initially receives a WinnerNFT carrying a 10 AVAX cycle-specific claim right. The holder can choose among several paths:

- Hold the WinnerNFT. No payment is taken, and the 10 AVAX claim right remains reserved behind that token.
- Transfer the WinnerNFT to another Avalanche-compatible address. The receiving address becomes the current holder and inherits the same unclaimed claim right.
- Claim 5 AVAX, which is 50% of the cycle target, while keeping the WinnerNFT. The NFT then carries only the remaining 5 AVAX claim right.
- Fully redeem the NFT. After approving FundingManager, the holder receives the remaining reserved balance and the NFT is transferred to the project wallet. If no partial claim occurred, that payment is 10 AVAX; after the 5 AVAX partial claim above, it is 5 AVAX.

This design separates winner selection from eventual redemption. Selection determines the initial owner. ERC-721 ownership determines who controls the remaining claim at any later time, and the claim ledger determines how much value remains available for that specific cycle token.

7. Generative artwork lifecycle

The live visualizer is part of the protocol experience but not part of Solidity consensus. It is an open-ended real-time generative system rather than a fixed preauthored collection. In the current implementation it can produce up to 75 unique candidate visual states per second on a 480x320 canvas that participants can see in both the Standard and Degen interfaces. These visual states are candidates, not minted tokens: when a cycle

completes, the oracle captures one current state before selecting the winner, and that captured frame becomes the completed cycle's WinnerNFT artwork.

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. If the completed cycle contains N participating addresses and the random index is uniform, each address occupies one of N selection positions. Contribution size does not change an address's number of entries. Its only indirect relationship to shared odds is through the fixed-capacity behavior described in Section 3: larger accepted contributions can leave less room for additional addresses before completion, while smaller contributions can allow a larger final participant set. This is explicitly an address-level model; the oracle and contract do not attempt to infer whether multiple addresses share one controller.

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. Address-level participation itself is intentional; the remaining mainnet question is whether low-value address splitting needs a minimum economic cost or other entry rule. A production launch requires documented randomness and address-entry policies.

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.
Address-entry economics	One entry per participating address is intentional. Multiple controlled addresses can create multiple entries; because Fuji accepts any positive contribution, low-value splitting can bypass the full-cap concentration tradeoff. A mainnet entry-cost policy remains open.
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.
Governance	Governance is voucher-weighted, not person-weighted. Large voucher holders can hold substantial influence; votes permanently consume vouchers; ranked tallying may require multiple permissionless transactions; and exact ties can require additional voucher-funded runoffs. Independent audit and adversarial governance testing are required before production use.

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
Voucher burn endpoint	Zero address in this reference snapshot; separate governance deployment configures SwagBallGovernance as the burn-compatibility endpoint.
Public environment	fuji.avax.swagball.io
Deployment block	58,289,827
SwagBallGovernance	Not deployed in this reference snapshot; separate deployment module is implemented for Fuji testing.

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, including the separate governance contracts, voucher-burn compatibility, ranked tally batching, runoff behavior, and governance-weight concentration.
3. Choose and implement production winner randomness with a documented trust model.
4. Finalize the mainnet address-entry economics, including whether a minimum accepted contribution or other entry-cost rule is needed to keep low-value address splitting from bypassing the intended concentration tradeoff; test the resulting policy 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.
- Selection odds depend on the final number of participating addresses in a completed cycle. Contribution size can influence that participant count through fixed cycle capacity, but it does not weight an address or create extra entries.
- Address-level participation is intentional and does not identify humans. Multiple controlled addresses can create multiple entries; on Fuji, any positive accepted contribution can create an entry, so low-value address splitting can bypass the full-cap concentration penalty.
- Current randomness depends on an authorized off-chain oracle.
- The visualizer and HTTP storage path are operational dependencies outside the EVM.

The 75-state-per-second figure describes current visual-generator throughput, not NFT mint throughput, guaranteed uniqueness across all time, market value, or a protocol payout rate. WinnerNFT minting remains tied to completed funding cycles.

- Administrative powers remain meaningful and require stronger production controls.
- Governance weight is voucher-weighted rather than person-weighted. One controller may accumulate or acquire many unspent ParticipationVouchers and therefore hold substantial seconding or voting power. The design does not attempt proof-of-personhood.
- Binding governance consumes ParticipationVouchers permanently. Ranked tallying is permissionlessly advanced in bounded batches, and unresolved exact ties require additional voucher-funded runoffs; low participation or repeated ties can therefore delay a final decision.
- 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 image is public, but the permission is not: current NFT ownership is the transferable key that the FundingManager recognizes for the cycle's unclaimed reserve.

The project’s strongest architectural property is separation of concerns. Solidity protects cycle accounting and claims. Governance is isolated in a separate contract that can read cycle context without controlling funding reserves or winner 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 address-entry economics, 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.
createPetition(...)	Create a public governance petition; snapshot current cycle as its permanent second quota and store the petition hash/metadata/options.
secondPetition(motionId, voucherIds)	Add one non-destructive seconding unit for each currently owned voucher that has not already seconded that petition.
vote(motionId, voucherIds, ranking)	Burn approved ParticipationVouchers and create one weighted ballot plus one locked VoteReceiptNFT for

	the voting wallet.
startTally / processTally	Begin the post-deadline count and permissionlessly advance ranked-choice eliminations/ballot transfers in bounded batches.
createRunoff / applyRunoff	Resolve exact historical elimination ties with a fresh voucher-funded runoff rather than randomness; advance the runoff winner in the parent tally.

Key events

- ContributionReceived / ContributionRefunded
- CycleCompleted
- ProjectPaid / ProjectPaymentAccrued / AccruedProjectFundsWithdrawn
- RandomWinnerSelected
- PartialFundsClaimed / FundsClaimed
- PhasesScheduled / PhasesActivated / PendingPhasesCancelled
- OracleUpdated / ProjectWalletUpdated
- ContributionsPaused / ContributionsUnpaused
- SurplusRecovered / MigrationFinalized
- PetitionCreated / VoucherSeconded / MotionQualified / PetitionExpired
- VoteCast / TallyStarted / OptionEliminated / BallotWeightTransferred / BallotWeightExhausted
- RunoffRequired / RunoffCreated / RunoffApplied / MotionFinalized / MotionNoDecision

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 funding round with a snapshotted target, 120% completion threshold, and per-wallet cap. It has no fixed countdown; it remains open until the completion threshold is reached.
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.
ParticipationVoucher / VoucherNFT	First-contribution ERC-721 for a cycle. While unspent, each voucher is one unit of seconding/voting capacity; seconding is reusable, while binding voting permanently burns the voucher.
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.
Motion	A petition that reached its snapshotted voucher-weighted second quota and entered binding voting.
VoteReceiptNFT	Non-transferable NFT minted to a voting wallet after ParticipationVouchers are burned. It records the motion, voter, weight, and ranking and carries no governance power.
MotionRecordNFT	Non-transferable archive NFT minted to the project wallet when a motion is finalized. It carries no governance power.
Runoff	A new voucher-funded vote among exactly tied options when prior-round history cannot deterministically identify an elimination. Randomness is not used for governance ties.

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. An unspent ParticipationVoucher is also one unit of governance weight: it can second petitions without being consumed, and it can later be permanently burned for one unit of binding voting weight.

How does a public petition become a governance vote?

A petition snapshots the current funding-cycle number as its required seconding quota. Current ParticipationVoucher holders contribute one second per voucher without consuming those vouchers. Once the quota is reached, the petition becomes a qualified motion and a seven-day voting window opens. If the quota is not reached, the petition follows the seven-day/three-cycle expiration rule with a 30-day hard stop.

How do ParticipationVouchers become votes?

Only ParticipationVouchers are exchanged for voting weight. A wallet chooses how many vouchers to commit, approves them to SwagBallGovernance, and submits one ballot. The committed vouchers are permanently burned, and each burned voucher contributes one unit of ballot weight. VoteReceiptNFT and MotionRecordNFT are historical records only and cannot be recycled into governance weight.

How are motions with more than two outcomes decided?

They use weighted ranked-choice voting. A ballot's voucher-derived weight starts with its first preference and moves to the next surviving ranked preference when an option is eliminated. Exact unresolved elimination ties do not use randomness; they require a new voucher-funded runoff among the tied options.

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?

Not directly under the current Fuji rules. Each participating address appears once in the selection set regardless of contribution amount, so contributing more does not buy additional entries or directly weight that address more heavily. Contribution size can affect shared odds only indirectly by consuming fixed cycle capacity: larger accepted amounts can leave less room for additional addresses to enter before completion, while smaller amounts can allow more addresses into the same cycle. At selection time, every participating address still occupies one position in the final set. The unit is the address, not a verified human identity.

Why would someone contribute more if contribution size is not weighted?

Because a cycle has limited funding capacity. A larger accepted contribution occupies more of that capacity and can leave less room for additional wallets to enter before the cycle completes. That may indirectly reduce the final participant count, but the contributing wallet still receives only one selection position.

Can one person participate with multiple wallets?

Yes. SwagBall counts participating addresses and does not attempt to prove that one wallet equals one human. Each participating address receives one selection position. The 20% cap limits how much one address can fund; it does not limit how many addresses a person may control.

Does using more wallets guarantee a better economic outcome?

No. When multiple controlled addresses are used to occupy more of the cycle's funding capacity, the controller puts more aggregate capital at risk while the winner reserve remains fixed at the cycle target. Four full-cap controlled addresses contribute 80% of target for four of six positions in a six-address full-cap cycle. Six full-cap controlled addresses can guarantee the selected address by funding the entire 120% cycle, but full WinnerNFT redemption returns at most 100% of target, creating a guaranteed 20% target-sized net funding cost before gas. Fuji still allows very small accepted contributions to create entries, so low-value address splitting remains a separate mainnet design consideration.

How long does a funding cycle stay open?

There is no fixed countdown in the current design. The funding capacity determines the duration: the cycle remains open until accepted contributions reach 120% of target. A busy cycle can complete in seconds; a slower cycle can remain open for days, weeks, or longer. Winner selection occurs only after the cycle completes.

What is the difference between ParticipationVoucher and WinnerNFT?

ParticipationVoucher records first participation in a cycle and can serve as governance weight while it remains unspent. WinnerNFT is minted to the selected participant after a completed cycle and carries the remaining on-chain claim right for that cycle. ParticipationVoucher can be burned to cast governance weight; WinnerNFT ownership instead determines who is entitled to receive the still-unclaimed winner reserve.

Why use an NFT at all?

Because the WinnerNFT is not relying on image scarcity as its utility. The image can be copied, but the on-chain permission cannot. The NFT gives SwagBall a unique, transferable ownership object that FundingManager can verify and redeem. In the simplest model: the NFT is the key and the contract-controlled cycle reserve is the safe.

Does SwagBall mint 75 NFTs per second?

No. The live visual engine can produce up to 75 unique candidate visual states per second. WinnerNFT minting remains tied to completed funding cycles: one completion-state frame is captured and committed as that cycle's artwork. The generator is high-throughput and open-ended; the NFT is cycle-bound.

What gives a WinnerNFT utility at mint?

The utility is present when the token is minted. Current ownership already controls the cycle-specific remaining claim right backed by the winner reserve. Before any partial claim, that contract-defined claim can equal the full cycle target. This is an on-chain redemption right, not a guarantee that an independent market will price the NFT at the same amount.

Can the winner transfer a WinnerNFT?

Yes. Under the current ERC-721 design, the WinnerNFT can be transferred to another compatible address. The remaining claim right follows current NFT ownership, so the receiving address becomes the party able to use the available claim actions.

Does a WinnerNFT always have a market value equal to the cycle target?

No guaranteed secondary-market price is implied. The protocol-defined value is the token's remaining claim right against funds reserved for its cycle. Before any claim, that right can equal the cycle target. After a partial claim, only the unclaimed balance remains. Any independent market price for the NFT can differ from that contract-defined redemption amount.

How does a holder cash in a WinnerNFT?

For full redemption, the current holder approves FundingManager to transfer the WinnerNFT and then uses the full-claim process. The contract transfers the NFT to the project wallet and pays the holder the remaining reserved balance for that token's cycle. Full redemption closes the claim; no additional winner reserve remains available for that NFT.

Can a holder take part of the claim and keep the NFT?

Yes. The current holder may claim up to 50% of the cycle target without surrendering the NFT. The token remains transferable afterward, but only the unclaimed balance follows it to a later owner.

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