Multiplier: The People's Market Maker

July 8, 2025

Abstract

Multiplier is a behavioural-liquidity protocol that transforms gameplay into structured token demand. Through onchain games and an auction-driven curation layer, Multiplier routes capital into curated tokens based on player outcomes and community coordination—creating a self-reinforcing loop of attention, narrative momentum, and buy pressure.

Token projects earn exposure not via emissions but by winning slots in a dynamic, stake-based auction game, while developers integrate seamlessly via SDKs to build custom game experiences on shared liquidity.

This fusion of play, probability, and market mechanics converts passive speculation into active engagement, reactivating stalling tokens and sustaining liquidity beyond initial launches. Multiplier's modular architecture and probabilistic design primitives establish a new micromarket structure primitive where entertainment drives markets.

Contents

Executive Summary

Problem:

Despite billions in emissions and complex incentive models, most protocols overlook the real constraint in onchain markets: behaviour. Capital is abundant. Attention is not. Multiplier corrects this by embedding variance-driven gameplay into liquidity routing, turning user activity into structured token demand.

Gamification as Market Infrastructure:

Multiplier transforms liquidity into a game surface. Players engage for dopamine, upside, and leaderboard visibility. Token communities coordinate to secure exposure through staking. Developers integrate games via SDKs to monetize attention and tap into protocol volume. Every role contributes to, and extracts from, the same system via gameplay.

Core Mechanism:

Each win routes capital into a curated token. Curation is governed by auction-based slot mechanics, exposure is earned, not subsidized. Tokens gain volume not through emissions, but because gameplay drives market interaction.

Curation and Coordination:

Slots decay daily, forcing continuous coordination. Each \$GAMBA stake burns 5%, embedding economic commitment. Tokens rise and fall not by emissions, but by narrative velocity, staking alignment, and gameplay performance. Buy pressure becomes earned, visible, and reflexive.

Architecture:

Modular smart contracts decouple gameplay, routing, execution, and risk layers. Developers can deploy games, onboard tokens, or fork core logic with minimal friction.

Vision:

Multiplier protocol introduces a new liquidity layer, where speculation becomes structured demand and gameplay volume is routed through into markets. Multiplier is not a protocol with games. It is a gamification layer where liquidity is routed not through incentives, but through entertainment, coordination, and play.

1 Introduction

1.1 Context

Abundant yet fragmented capital, rapid experimentation, and a paradoxical scarcity of attention characterize today's decentralized markets. While token creation and smart contract development are easier than ever, the effective coordination of demand remains elusive. Users are inundated with projects, protocols, and narratives, each vying for liquidity yet few possess reliable mechanisms to route and concentrate that liquidity with intent.

Existing onchain infrastructure leans heavily on financial orthodoxy: liquidity mining, emissions schedules, staking programs, and centralized market makers. These tools can bootstrap initial volume, but often degrade into extractive equilibria. Incentives are farmed, liquidity is rented, and user engagement dissipates once rewards expire.

This exposes a fundamental design limitation: DeFi's basic building blocks excel at attracting capital but fail to create sustainable patterns of user engagement and behaviour.

1.2 The Premise

Multiplier proposes a reframing:

Rather than treating users solely as transaction agents, Multiplier models user activity as probabilistic input within a reflexive system, where each action contributes to dynamic market topology shaped by attention, variance, and feedback loops.

Through a calibrated system of games, curation auctions, and onchain execution, high-variance user actions are converted into liquidity injections. When a user wins, the protocol routes capital into a curated token. These tokens are selected via auction, ensuring that attention—not just capital—is priced and allocated.

The outcome is not just engagement, but coordination.

1.3 The Design Leap

Where traditional DeFi protocols attempt to simulate user participation through incentives, Multiplier generates real participation through gameplay.

The user is not asked to stake or trade, they are just asked to play.

And in doing so, the user injects liquidity, moves prices, shifts attention curves, and recursively participates in narrative formation. This is not gamification as surface-level UX.

It is gamification as a routing primitive.

1.4 Why It Matters

This inversion yields several first order effects:

- **Liquidity becomes reactive**, responding to player variance rather than strategic liquidity provision.
- **Token prices become reflexive**, shaped by in-game outcomes and social momentum, not just AMM mechanics.
- **Slot access becomes economically coherent**, as token teams bid for routing rights based on observable volume, not speculative TVL.
- **Behaviour becomes legible**, producing data-rich loops for optimization and extension.

In short, the system reorients infrastructure around behaviour as a service.

2 Behavioural and Market Foundations

2.1 Yield is Not Utility

Contemporary DeFi ecosystems are dominated by yield primitives: staking, farming, and bonding. These are financial abstractions presented as user incentives. While they satisfy rationalist frameworks, they fail to engage users at the behavioural layer. They offer **expected value**, but not **convex experience**.

Utility, in practice, is non-linear.

Let user utility $U(x)$ be a function of reward x, incorporating second-order curvature:

$$U(x) = \mathbb{E}[x] + \frac{\sigma^2}{2} \cdot \frac{d^2U}{dx^2}$$

Where:

To reason about user behaviour in probabilistic environments, we define:

- $\mathbb{E}[x]$ is the expected monetary return
- σ^2 is the variance of the payout
- $\frac{d^2U}{dx^2}$ is the curvature of the utility function (risk preference)

In recreational and bounded contexts, users exhibit **variance seeking** behaviour. They prefer volatility, not in spite of the risk, but because of the emotional asymmetry it produces.

Standard expected utility theory is insufficient. A more accurate model follows prospect theory:

$$U(x) = \begin{cases} x^\alpha & \text{if } x \geq 0 \\ -\lambda(-x)^\beta & \text{if } x < 0 \end{cases}$$

Where:

- $\alpha, \beta \in (0, 1)$ capture diminishing sensitivity
- $\lambda > 1$ represents loss aversion

Users also distort probabilities. The weighting function is:

$$w(p) = \frac{p^\gamma}{(p^\gamma + (1-p)^\gamma)^{1/\gamma}}, \quad 0 < \gamma < 1$$

The resulting decision utility is given by:

$$U_{\text{decision}} = \sum w(p_i) \cdot u(x_i)$$

This explains why staking at 4% APY is rational, but uninspiring, while high-variance games generate intense engagement per unit of capital, which have been proven by the success of systems like pump.fun. The key isn't expected value, it's shaped variance within bounded contexts.

2.2 The \$100K–\$10M Reflexivity Window

Most tokens don't fail at launch. They fail in middle of the lifecycle.

Between \$100K and \$10M market cap, tokens enter a zone where narrative decays, market makers exit, and liquidity thins. The reflexive loop between price, attention, and participation breaks. Multiplier targets that band directly.

Each win routes capital into a curated token, producing real market impact and narrative reinforcement. Tokens that would otherwise stall are reactivated by gameplay-induced buy pressure. This is not yield farming. It is **reflexivity farming**.

2.3 Gameplay as Capital Coordination

Markets are coordination systems. DeFi protocols simulate coordination through emissions and yield schedules, assuming rational actors and infinite attention. But users operate under session-based behavior, social signaling, and bounded cognition. Gameplay provides a native coordination loop:

- Short event intervals (1–3 minutes)
- Clear token-linked outcomes (wins inject liquidity)
- Immediate feedback (price movement, leaderboard updates)
- Social visibility (jackpots, screenshots, memes)

This converts attention into structured economic impact without the overhead of planning or passive staking.

2.4 The Meta-Narrative: Play-to-Inject

Multiplier's infrastructure does not recreate *play-to-earn*. It defines *play-to-inject*.

- **Play to earn**: Users extract value from the system
- **Play to inject**: Users become demand side participants routing capital and narrative via gameplay

Each win triggers execution. Each user becomes a micro liquidity router. The difference is not semantics it's systemic.

2.5 Elasticity of Attention

At small market caps, attention is both scarce and highly elastic. A $5K gameplay driven buy can move a token by 20%, not only due to liquidity constraints, but because of the narrative weight it carries (screenshots, leaderboards, social signaling).

This dynamic is captured simply as:

$$\frac{dP}{dA} \gg 1 \quad \text{for small market cap tokens}$$

Where P is price and A is attention. The marginal impact of attention on price grows non-linearly in illiquid conditions.
This creates a **reflexive amplification zone**:

- Price responds to attention
- Attention is driven by gameplay
- Gameplay routes capital
- Capital reinforces price

Multiplier's protocol is built to operate inside this feedback loop.

3 Protocol Design and Liquidity Mechanics

3.1 Pool-Based Execution: From Play to Token Buy

At the heart of Multiplier's system is a deterministic contract execution pathway governed by probabilistic user outcomes. Each user interaction involves committing capital toward a structured play, which upon resolution triggers a token buy from a shared pool. Multiplier's system enforces a separation between randomness and execution: while outcomes are randomized, execution is deterministic and auditable.

The outcome surface defines the set of possible reward tiers. Once a play is resolved by the backend and signed, the result is verified onchain, and the corresponding deterministic payout is triggered—executing a token buy from the shared liquidity pool.

This ensures that each action taken by a user results in a predictable liquidity event, governed by probabilities and risk surfaces. Rather than routing capital passively into markets, the protocol allows users to express intent and face probabilistic outcomes whose resolutions directly alter the size and distribution of the buy-side pressure.

3.2 Outcome Surfaces and Tier Structures

Each pool can be configured with custom **outcomes**: Number of tiers, Payout percentages, Selection and win probabilities

Multiplier's pool structure introduces variance into the user experience, allowing developers to shape the economic topology of onchain interactions.

Tier structures may include any spectrum of design:

- **High frequency, low magnitude wins**
- **Medium-risk volatility curves**
- **Ultra convex jackpot style payouts**

Multiplier's modular design enables market participants to tune economic behaviour toward different objectives: long-tail engagement, high-throughput gameplay, or liquidity shock targeting. The outcome surface becomes a programmable layer of behavioural incentives, shaping how capital flows and when it concentrates.

3.3 Expected Value and Volatility Design

Each configured pool is fully transparent onchain. The core of its design lies in three parameters:

- **Expected Payout**: $E[P]$ - the mean reward given a win
- **Expected Consolation**: $E[M]$ - the mean loss-bounded reward given a miss
- **Volatility**: The degree of dispersion between outcomes across plays

We define:

- S_n: Pool size after n plays
- c: Cost to open a pool-based play (net of fees)
- $P(s_i)$, $P(w_i)$: Probability of selecting, winning tier i
- p_i: Payout % of S_n if tier i is won
- m: Consolation % of S_n if tier i is lost

The expected value of a single play is given by:

$$\mathbb{E}[P] = \sum_{i=0}^{T} P(s_i) \cdot P(w_i) \cdot p_i$$

And the expected consolation value when the tier is lost is:

$$\mathbb{E}[M] = \sum_{i=0}^{T} P(s_i) \cdot (1 - P(w_i)) \cdot m$$

3.4 Multi-Tier, Multi-Pool Mathematical Formulation

The pool size after each round is affected by both the expected gain from the user's payment and the expected loss from the pool's payout. As plays accumulate, the shared liquidity pool evolves based on inflows (wagers) and outflows (payouts). The recursive update formula is:

$$S_n = S_{n-1} \cdot (1 - \mathbb{E}[P] - \mathbb{E}[M]) + c$$

Where:

- S_n: Pool size after n plays
- $\mathbb{E}[P]$ and $\mathbb{E}[M]$: Expected win and consolation payout

Over time, the system converges toward:

$$\lim_{n \to \infty} S_n = \frac{c}{\mathbb{E}[P] + \mathbb{E}[M]}$$

This provides a mathematical guarantee that the pool will stabilize under probabilistic constraints, allowing developers to simulate and test behaviour before deployment.

3.5 Equilibrium Modelling: Convergence and Risk Boundaries

To model stability, Multiplier tracks each pool. As user interactions increase, the pool moves towards a defined equilibrium based on:

- The fixed capital injection c
- The probabilistic drawdowns from payouts and consolations
- The relative size of the payouts (as a % of pool)

Therefore, each play contributes more to the pool than it extracts, allowing long-run convergence toward a finite pool size. The rate of convergence depends on how close the total expected outflow is to 1. In this case, pool depletion slows geometrically and equilibrium is guaranteed. If it exceeds 1, the system is unsustainable and requires parameter rebalancing.

3.6 Dynamic RTP, Game Flexibility, and Configuration Tuning

RTP (Return to Player) and risk exposure can be tuned dynamically before launching a game:

- Adjusting tier weightings $P(s_i)$, tier win chances $P(w_i)$, or payout sizes p_i
- Configuring tier-based promotional campaigns or liquidity shock events
- Introducing dynamic variables based on pool performance

Each configuration produces a predictable and measurable change in system behaviour, empowering developers to fine-tune user engagement loops and liquidity behaviours with mathematical precision. From conservative staking simulations to high-volatility meme surface games, the infrastructure supports the full spectrum of onchain behaviour.

4 System Architecture

4.1 Architectural Premise (Overview)

The Multiplier system orchestrates wagering into liquidity routing protocol using a layered execution pipeline. Multiplier's system architecture is built to handle high-throughput probabilistic interactions across multiple game configurations, while simultaneously preserving deterministic outcomes and just-in-time onchain execution. The system routes user activity through a tightly integrated pipeline:

- A **contract layer** that registers player intents and organizes resolution queues
- An **off-chain backend** that processes game logic and computes signed outcomes
- A **payout module** that ensures capital-efficient token delivery from protocol-controlled treasury wallets

This setup enables composable and cost-efficient liquidity activation at scale. Importantly, Multiplier separates resolution logic from contract state, which unlocks the ability to run fast-paced game loops. The system can be decomposed into three primary flows:

1. **User intent and wager submission**
2. **Off-chain resolution and result signing**
3. **Onchain payout execution**

Multiplier's protocol architecture allows:

- Deterministic outcomes with cryptographic signing
- Payout queues and attempt resolution are transparently managed
- Just-in-time treasury settlement with minimal latency
- Pluggable frontend and backend SDKs for ecosystem games

4.2 Contract Layer: Registering Intents & Managing Payout Queues

Users initiate play by submitting an intent to Multiplier's game contract. Each intent:

- Specifies wager amount c, selected pool, and optional token routing
- Enters a **payout queue** indexed by game and ordered by timestamp (on chain)
- Receives a unique identifier used to validate off-chain resolution

Multiplier's game contract acts as a coordination layer. Its primary functions are:

- Accept player intents
- Connect a unique identifier for downstream verification
- Hold a payout queue, grouped by game and ordered by timestamp, that waits for verified results

Multiplier's queue guarantees that resolutions must match user-registered parameters. This separation between registration and resolution reduces gas cost and increases platform scalability.

4.3 Game Backend: Attempt Resolution & Result Storage

Once intent is submitted, the resolution engine:

- Listens for onchain events, containing the wager and context
- Uses a pseudo-random number seed (derived from context and wager [Similar to a VRF]) to evaluate the outcome
- Applies the payout function defined in the lootbox pool (see Section 3)

Each outcome tier i has parameters:

- Tier probability t_{ik}
- Win chance u_{ik}
- Payout ratio p_{ik}

The backend computes result r:

- Outcome tier i
- Win or consolation flag
- Payout % of pool

This payload is then submitted onchain to the payout contract to execute resolution.

4.4 Payout Pipeline: JIT Settlement from Treasury Wallets

The payout pipeline is responsible for delivering token proceeds directly to users based on the signed result.

Key mechanics:

- Each resolved play triggers a **just-in-time (JIT)** payout event
- A treasury wallet managed by protocol governance holds payout reserves
- Upon verified resolution, funds are transferred to the player's wallet in the correct token and amount

This model ensures that each payout is fully covered by play capital and treasury provisions, while keeping idle capital off-chain for optimal capital efficiency.

4.5 SDKs and Studio Onboarding Infrastructure

Multiplier's system architecture deliberately separates outcome determination from frontend design. Multiplier is designed as a backend infrastructure layer, enabling developers to build their own fully customized games. The protocol exposes a clean SDK layer that handles resolution, execution, and payout orchestration, while allowing external teams to construct game UX, interfaces, and additional mechanics of their own choosing.

The developer-facing SDK abstracts away all core logic and provides a clean interface for:

- Registering user intents with selected pool/game configurations
- Submitting signed outcome resolutions for onchain execution
- Querying session and payout status

Developers retain full creative freedom over the front-end experience: whether it's a spin wheel, PvP contest, or animated jackpot—while Multiplier handles:

- Deterministic contract routing
- Probabilistic resolution logic
- Just-in-time payout execution

The onboarding infrastructure includes:

- Frontend-ready SDKs and hooks for wallet interactions and play sessions
- Tools to define pool configurations:
  - Outcome tiers and volatility settings
  - Payout multipliers (p_{ik}), win chances (u_{ik}), and selection weights (t_{ik})
- Templates for safe backend signing of outcome resolutions

This model empowers ecosystem studios to launch diverse and high-performance game experiences using Multiplier as the foundational execution layer.

Additionally all write-level interactions with protocol contracts via the SDK are permissioned and require $GAMBA, ensuring aligned usage and economic consistency across developer integrations. Studios integrating via Multiplier's SDK gain embedded access to Multiplier's shared liquidity, slot curation system, and fee capture pathways—providing direct revenue upside in exchange for routing gameplay volume through the protocol.

4.6 Chain-Agnostic Design and Future-Proofing

The system is intentionally built with **cross-chain composability** and future-proof primitives. Key design choices include:

- Stateless contracts for play validation
- Portable surface definitions
- Signature-based resolution logic (compatible with any smart contract system)

This allows the same backend to serve multiple frontends and chains, and even support L2s and non-EVMs. Pool accounting, payout logic, and resolution pathways are modular and forward-compatible.

This ensures the core logic of "play $\rightarrow$ resolve $\rightarrow$ buy" remains constant while enabling ongoing enhancements in speed, scalability, and interoperability.

5 Coordination Layer and Emergent Dynamics

At the heart of the Multiplier protocol lies a novel coordination mechanism: the curation layer. This layer determines how buy pressure is routed, which tokens receive it, and how sustained community engagement shapes access to platform volume. It unifies token curation, staking, and liquidity routing into a single probabilistic system governed by $GAMBA.

5.1 $GAMBA Utility: Routing, Boosting, Staking

Every action that influences token visibility, gameplay routing, or protocol yield flows through $GAMBA. Users stake it to boost tokens into curated slots. Developers consume it to access backend infrastructure. And the system redistributes it through yield cycles tied to real platform usage.

The utility triangle of $GAMBA:

- **Curation** → Boost token ranks for gameplay routing and exposure by staking $GAMBA
- **Staking** → Earn protocol rewards (SOL/USDC) after 30-day maturity
- **Execution** → Required for SDK infrastructure calls and settlement logic

By collapsing staking, curation, and infrastructure payment into a single token, the system avoids fragmented incentives. $GAMBA becomes the single signal for capital commitment, economic exposure, and execution demand.

5.2 Curation Slot Auctions: Bidding for Buy Pressure

The curation system is a live auction for attention. Users stake $GAMBA on any token at any time. Each $GAMBA staked equals **1 Curation Point (CP)**.

The top 10 tokens by CP enter active slots. These slots are updated **dynamically**, meaning:

- No rounds
- No voting periods
- No cooldowns

Buy pressure from non-user-selected tokens is routed across these slots probabilistically, with higher ranks getting larger shares. This turns attention into measurable economic impact.

Each new staking action burns 5% of $GAMBA, creating embedded economic commitment with each coordination attempt.

5.3 Incentive Alignment Across Players, Token Communities, Developers, and Stakers

The protocol aligns core stakeholders by rewarding active contribution to the system's growth and liquidity flow:

- **Players** participate for direct upside: engaging games, token exposure, and payout potential.
- **Token communities** coordinate $GAMBA staking to gain visibility and receive buy pressure, turning attention into market impact.
- **Developers** integrate via SDK and earn from gameplay volume — either through platform share or stand-alone infrastructure usage.

- **Stakers** commit \$GAMBA to curated tokens, influencing visibility and earning protocol yield in SOL/USDC.

The result is a competitive yet collaborative loop where each group amplifies the others — routing more volume, sustaining more attention, and deepening economic alignment across the ecosystem.

5.4 Narrative Rotation and Dynamic Slot Equilibrium

Slot dominance is not permanent. Instead, tokens face **position-based daily decay**:

Let C_t be the Curation Points at day t. For token in slot k with decay rate d_k, we get:

$$C_{t+1} = C_t \cdot (1 - d_k) \tag{1}$$

For example, the decay rates could be initially set as

$$d_1 = 0.08, d_2 = 0.07, ..., d_{10} = 0.015$$

This creates a gravity-like pull: tokens need continuous support to maintain rank, or else decay pushes them out.

Communities must actively coordinate to keep tokens visible. This introduces an always-on, strategy-rich curation meta.

This means whales can't stake indefinitely if they want to keep a high token position.

5.5 Recursive Liquidity and Attractor Theory

Gameplay → Liquidity Routing → Token Price Impact → More Staking → Higher Slot Position → More Gameplay

This reflexive loop forms **dynamic attractors** in the system: tokens that start gaining attention and price momentum can rapidly concentrate routing flow — unless actively challenged by other communities.

When a user initiates a game but **does not manually select a token**, Multiplier routes the resulting buy pressure according to the **curated slot rankings**.

Let S_i denote the slot, $i \in (1, 10)$, and R_i the routing weight for that slot. Default routing follows a **graduated distribution curve**, rewarding top slots while preserving competitive viability for others.

Total Routing:

$$\sum_{i=1}^{10} R_i = 1 \tag{2}$$

- **Manual token selection overrides this routing.** Any token, even non-curated ones, can receive gameplay-generated liquidity.
- By default, however, most volume is routed through this curated layer — creating a public attractor state for tokens that coordinate effectively.

This mechanism builds reflexivity:

As communities rally to stake, they push tokens into visibility → Which increases gameplay-driven volume → Which reinforces further staking.

Let L_k be the proportion of liquidity routed to slot k, and assume a weighting curve $W(k)$:

$$L_k = \frac{W(k)}{\sum_{j=1}^{10} W(j)} \tag{3}$$

The shape of $W(k)$ can be adjusted, for example it could be exponential or linear, this can be used to tune concentration vs distribution across slots.

This makes routing programmable, and attractors observable. Developers and protocol designers can simulate outcomes based on different auction curves and decay rates.

5.6 The Behavioural Meta: Gamified Liquidity as UX Primitive

At its core, Multiplier transforms liquidity routing from a backend optimization problem into a visible, participatory game. Every element of the system from curation slots to decay mechanics is designed to project internal capital dynamics into the user interface, making economic coordination feel like gameplay.

Liquidity as Game Surface

In traditional financial systems, liquidity flows are opaque. In Multiplier, they are surfaced, shaped, and fought over. The curated slot leaderboard is not just a staking outcome—it's a public battleground. Players can see, in real time, which tokens are rising, which tokens are decaying, and how their own contributions change the outcome.

Strategy Loops and Emergent Incentives

Because slot positions decay over time, Multiplier introduces **an always-on demand for strategic re-engagement**. Communities can't simply stake and forget, they must decide *when*, *where*, and *how much* to stake in response to other actors. This creates a set of emergent strategy loops:

- **Time-based Coordination:** Communities must synchronize staking behaviour to maintain or regain positions.
- **Reactive Liquidity Boosting:** Sudden losses in ranking prompt urgent action, encouraging fast-moving collective decisions.
- **Slot Sniping:** Actors wait until decay weakens a slot leader, then inject just enough $GAMBA to overtake it.
- **Passive Play vs. Active Defense:** Top tokens must balance holding their lead with the cost of overcommitment.

This is not governance staking, it's *real-time game theory* played out in economic terms.

Reflexive UX and Market Impact

Multiplier's behavioural layer creates what can be described as a **reflexive UX**: the system incentivizes behaviours that, in turn, reshape the very environment users are responding to.

- When a user stakes $GAMBA, they increase CP and improve token rank
- That higher rank routes more gameplay-driven liquidity to their token
- That volume impacts market price and visibility
- Which makes further staking rational and likely
- Which strengthens their rank—until another community counters

From Incentives to Identity

Over time, the behavioural meta begins to form **identity structures**. Players align themselves with tokens not just for speculative upside, but for status, reputation, and group belonging. Token communities evolve into **factions**, coordinating in Discords, sharing dashboards, rallying for slot takeovers.

6 Conclusion

Multiplier is more than a protocol. It's the foundation of a new economic primitive where play drives purpose. By aligning probabilistic gameplay with onchain markets, it creates a self-sustaining loop of attention, narrative momentum, and buy pressure that extends well beyond token launches.

Our vision for Multiplier is that it embeds itself into existing loops and UX layers: silent in the background yet alive in every click. Through dynamic auctions, slot decay, and seamless integration, the protocol quietly fuels continuous engagement, ensuring markets stay vibrant, tokens remain vital, and communities always have something to play for.