SpaceWheat Quantum Architecture Proposal

Version 1.1 | January 2026 | Revised Draft

Executive Summary

SpaceWheat is a farming simulation game built on authentic quantum mechanical principles. Players tend quantum gardens by adjusting physical parameters, applying quantum gates, creating entanglement, and performing measurements. The game aims to teach quantum mechanics intuitively through gameplay rather than abstraction.

This proposal defines a unified architecture for the quantum simulation engine. The core design separates emoji-based physics definitions from Hilbert space geometry, allowing diverse biome types while maintaining mathematical rigor. Biomes operate as isolated quantum sandboxes with no cross-biome entanglement, enabling scalable complexity without exponential resource growth.

The architecture supports two complementary gameplay modes: analog tuning of continuous physical parameters, and digital manipulation through discrete quantum gates. Both modes operate on the same underlying density matrix formalism, ensuring physical correctness.

Design Principles

Authentic Physics. The simulation implements the Lindblad master equation, the standard formalism for open quantum systems. Evolution preserves trace, Hermiticity, and positive semidefiniteness of the density matrix. Measurement follows the Born rule with proper state collapse.

Separation of Concerns. Emoji physics definitions (Icons) are independent of Hilbert space structure (Embeddings). The same Icon library applies across all biomes, with physics instantiated contextually based on which emojis exist in each biome.

Biome Isolation. Each biome is a self-contained quantum system. No entanglement exists between biomes. Resources flow between biomes classically, not quantum mechanically.

Flexible Geometry. Biomes may use qubit-pole structure (emojis label north/south poles of qubits, enabling gates and entanglement) or direct labeling (each emoji is a basis state). The physics engine handles both uniformly.

System Architecture

The system comprises five layers:

Layer 1: Icon Library

Icons define the physics of individual emojis independent of any particular biome. Each Icon specifies self-energy (diagonal Hamiltonian term), Hamiltonian couplings to other emojis (off-diagonal terms), Lindblad outgoing rates (irreversible population transfer), and optional time-dependent driving parameters. The Icon library is global via a central IconRegistry. This registry pattern enables game balance tuning: designers can adjust physics parameters in one location without modifying simulation code.

Layer 2: Emoji Embedding

An EmojiEmbedding maps emojis to basis state indices within a specific Hilbert space. Qubit-Pole Embedding: Each qubit axis has north (|0>) and south (|1>) emojis. For n qubits, dimension = 2^n. This enables quantum gates and entanglement. Direct Embedding: Each emoji is one basis state. For n emojis, dimension = n. No gates applicable. The EmojiEmbedding layer is architecturally critical as it translates between the player's visual world and underlying matrices, enabling moddability without touching simulation code.

Layer 3: Quantum System

Each biome owns one QuantumSystem instance, consolidating QuantumBath and QuantumComputer. Contains: density matrix (complete quantum state), Hamiltonian matrix, Lindblad jump operators, and the embedding. Evolution follows the Lindblad master equation. Measurement projects and renormalizes. Gates embed unitaries into full space.

Layer 4: Biome Definition

A BiomeDefinition resource specifies: name, structure type (qubit-pole or direct), axes or emoji list, initial state, and gameplay constraints. Example: Quantum Kitchen uses 3 qubits (temperature, moisture, grain/bread) with gates enabled. Forest Ecosystem uses 6 direct emojis with gates disabled.

Layer 5: Plots

Plots are measurement apparatuses, not state owners. Each references parent biome's QuantumSystem and specifies a measurement axis (north/south emoji). Planting registers the axis but does NOT perform measurement. The quantum state continues evolving freely. Only at harvest does projective measurement occur. This distinction avoids the Quantum Zeno Effect: planting is registration, not observation.

Player Toolkit

Analog Tools (continuous parameter adjustment):

Tune Coupling: Adjust Hamiltonian coupling strength between emojis. Tune Decay: Adjust Lindblad decay rates. Add Driver: Attach time-dependent driving fields (sinusoidal, pulsed). Pump: Incoherent population injection toward target emoji.

Digital Tools (discrete operations, qubit-pole only):

Single-qubit gates: H (superposition), X (bit flip), Z (phase flip), S, T (phase control). Two-qubit gates: CNOT (conditional flip), CZ (conditional phase), SWAP. Gate application prohibited on measured plots.

Quantum Actions (all biomes):

Entangle: Create Bell state between plots (qubit-pole only). Measure: Projective measurement with state collapse. Inspect: View probabilities without collapse. The Inspect tool is critical for direct embedding biomes where gates are unavailable. Recommended visualization: 2D heatmap of density matrix showing populations (diagonal) and coherences (off-diagonal).

State Steering Tools:

Weak Measurement: Partial observation that nudges state toward outcome without full collapse. Quantum Fertilizer: Narrows probability distribution around target emoji via engineered Lindblad operators that pump toward target while draining competitors. Preserves coherence for continued evolution. These tools address player frustration from measurement randomness.

Biome Isolation

Biomes are strictly isolated quantum systems. Benefits: Scalability (ten 8D biomes = ten 8x8 matrices, not 8^10). Semantic Clarity (same emoji can mean different things in different biomes). Gameplay Comprehension (learn QM in isolated sandboxes). Classical Resource Flow (biomes exchange classical tokens, not quantum states).

Performance Considerations

Dimension Limits: Matrix operations scale as O(d^3). Real-time 60 FPS feasible for d up to 32 (5 qubits). For d up to 256 (8 qubits), 10 Hz evolution acceptable. GDScript Optimization Path: ComplexMatrix performs heaviest computation. For 6+ qubit biomes, implement as GDExtension plugin (C++/Rust) for native performance while preserving GDScript API. Profile before optimizing. Independent Clusters: Unentangled qubits simulate as separate smaller systems until entanglement merges them.

Implementation Plan

Phase 1 - Core Refactoring: Consolidate QuantumBath + QuantumComputer into QuantumSystem. Implement EmojiEmbedding. Update builders. Deprecate: QuantumBath.gd, QuantumComputer.gd, DualEmojiQubit.gd.

Phase 2 - Biome Definitions: Create BiomeDefinition resource. Define 3-5 hand-crafted biomes. Validate Icon filtering and operator construction.

Phase 3 - Player Tools: Unified tool interface. Enable/disable gates by biome type. Implement analog tuning UI, measurement, inspection with density matrix heatmap visualization.

Phase 4 - Polish: Verify invariants under extended play. Profile performance. Validate edge cases.

Deferred Features

Player-Crafted Biomes: Architecture supports it; UX requires design work. Cross-Biome Quantum Effects: Initially excluded. Future 'Quantum Logistics' could implement teleportation via entangled portal plots across biomes, consuming entanglement as resource. Advanced Gate Sets: Fixed library suffices for beta.

Appendix A: Mathematical Specifications

Lindblad Master Equation: dp/dt = -i[H, p] + Sum_k gamma_k (L_k p L_k^dag - 1/2{L_k^dag L_k, p})

Cayley Method: U(dt) = (I - iHdt/2)^-1 (I + iHdt/2) preserves unitarity exactly.

Weak Measurement: p_after = (1-a)p + a*P_k*p*P_k/Tr(P_k*p) for strength a in [0,1].

Projective Measurement: P(k) = p[k,k]; p_after = |k><k|. For subspace S: P(S) = Sum_i p[i,i].

Appendix B: File Structure

```
Core/QuantumSubstrate/
  Complex.gd, ComplexMatrix.gd, DensityMatrix.gd
  QuantumSystem.gd (NEW), EmojiEmbedding.gd (NEW)
  QuantumEvolver.gd, HamiltonianBuilder.gd, LindbladBuilder.gd
  QuantumGateLibrary.gd, Icon.gd, IconRegistry.gd

Core/Environment/
  BiomeDefinition.gd (NEW), BiomeBase.gd
  [Specific biome implementations]

Core/Plots/
  BasePlot.gd
```