Technical Integration Review: Advanced Agent Behaviors ("The Complex 6")

Project: Universe Sandbox 2.0

Module: Evolutionary Logic & Agent Behavior

Status: Validated ✅

1. Executive Summary

I have conducted a comprehensive code review of the "Complex 6" update. This patch introduces a set of high-level agent behaviors—`MOVE`, `FORTIFY`, `HIBERNATE`, `DETONATE`, `TERRAFORM`, and `EMIT_LIGHT`—transforming the simulation from a static cellular automaton into a dynamic multi-agent system.

The integration logic successfully bridges the **Genotype** (mutation potential), the **Phenotype** (action execution), and the **Physics Engine** (state updates and metabolic costs).

2. Logic Analysis by Module

2.1. Mutation Engine (`innovate_rule`)

Implementation: Expansion of the `action_type` string pool.
Assessment: I confirmed that adding the new behavior strings to the random choice pool correctly enables the evolutionary discovery of these traits. The probability distribution remains balanced; with the expanded pool, high-impact actions like `DETONATE` remain statistically rare, acting as "special moves" rather than default behaviors.

2.2. Agent Decision Logic (`execute_action`)

Implementation: Execution blocks for new action types

Implementation: Execution blocks for new action types.
Assessment:

- **Locomotion (`MOVE`):** The logic correctly handles the atomic transfer of an object between grid coordinates. Deleting the dictionary key before re-insertion prevents state duplication ("ghost cells"). Mass-dependent energy scaling introduces a realistic physics constraint.
- **State Management (`FORTIFY` , `HIBERNATE`):** The use of the `state_vector` dictionary for temporary flags is efficient and avoids the need to restructure the core `OrganismCell` dataclass.
- **AoE Damage (`DETONATE`):** The targeting logic correctly identifies all Moore neighbors (8-way) regardless of kinship. The immediate self-pruning call guarantees the agent cannot act post-detonation, preventing logical paradoxes.
- **Environment Modification (`TERRAFORM` , `EMIT_LIGHT`):** Direct manipulation of grid cell properties (`temperature` , `light`) correctly persists changes to the shared environment, enabling niche construction and stigmergy.

2.3. Physics & Metabolism (`run_timestep`)

Implementation: Pre-computation checks for state flags.
Assessment:

- **Metabolic Suppression:** Placing the `HIBERNATE` check at the top of the loop is architecturally correct. It ensures hibernating cells bypass base metabolic costs, functionally simulating stasis.
- **State Reset:** The automatic reset of `is_fortified` ensures active defense requires a recurring energy investment, preventing "infinite armor" exploits.

3. Emergent Game Theory Analysis

3. Emergent Game Theory Analysis

Based on the implemented logic, I predict the following high-level strategies will emerge naturally from the evolutionary process:

Behavior	Predicted Emergence
The Turtle Strategy	Organisms combining high `energy_storage` with `FORTIFY` will create "bunker" colonies that outlast aggressive neighbors.
The Kiting Strategy	Organisms with `MOVE` and `sense_neighbor_other` will evolve to flee from `DETONATE` signals or high-density clusters.
The Dyson Loop	Colonial organisms will use inner cells with `EMIT_LIGHT` to sustain outer cells in low-light environments, creating self-sustaining metabolic loops.
Ecological Engineering	Lineages will use `TERRAFORM` to shift local temperatures to their specific `thermosynthesis` optimum, effectively creating a "home field advantage" that kills competitors.

4. Edge Case Validation

I have validated the logic against the following potential failure states:

- **Null Target Error:** The `MOVE` function includes a check for `empty_neighbors`. If a cell is completely surrounded, the action fails gracefully without crashing.
- **Energy Debt:** All actions check `cell.energy > cost` before execution. This prevents negative energy states that could break the simulation's conservation laws.
- **Target Validity:** The `DETONATE` logic uses the global lookup helper `get_target_at`, ensuring interaction with the global phenotype registry remains synchronized.

5. Conclusion

5. Conclusion

The "Complex 6" integration is architecturally sound. It adheres to the project's data structures and introduces no regression risks for existing behaviors. The logic is ready for deployment to the main branch.