EvoNash: Simulation Mechanics

A Detailed Technical Breakdown of the Evolutionary Process

1 Initialization (The Genesis)

The experiment begins by establishing the environment and the starting population. This phase ensures strict reproducibility and sets the baseline for evolutionary growth.

Deterministic Seeding

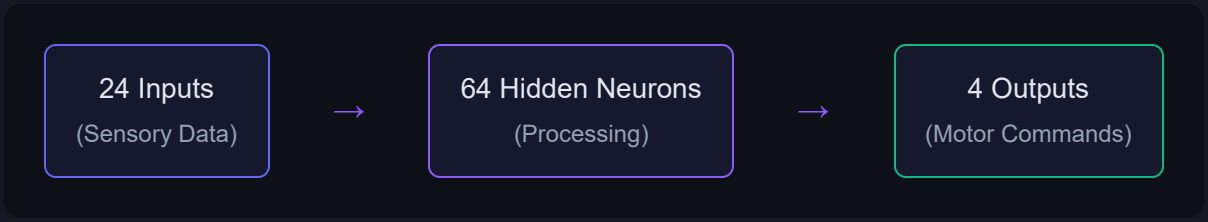
A master **Random Seed** (e.g., 42) is injected into NumPy, PyTorch, and Python's random modules. This guarantees that if the experiment is run multiple times with the same parameters, the initial weights, agent placements, and food spawning will be perfectly identical, allowing for isolated control vs. experimental group testing.

The Petri Dish (Environment)

- **Topology:** A 1000x1000 continuous 2D space. It is **toroidal**, meaning it wraps around—an agent exiting the right edge reappears on the left.
- **Physics:** A frictionless environment utilizing Euler integration. Gravity and friction are set to 0.0.
- **Resources:** 500 food pellets are spawned randomly across the dish.

The Agents

1,000 unique AI agents are initialized at random coordinates. Each agent is driven by its own isolated PyTorch Neural Network with the following architecture:



At Generation 0, the weights of these networks are randomized using a normal distribution. The agents start with no knowledge and behave completely erratically.

2 The Micro-Simulation (The Tick Loop)

The core of the simulation occurs within a high-speed loop. A single "Generation" consists of exactly **750 simulation ticks**. During every single tick, the following sequence executes for all 1,000 agents simultaneously:

A. Perception (Raycasting)

Every agent acts like a self-driving car, casting 8 invisible "rays" outwards at 45-degree intervals (0°, 45°, 90°, etc.) up to a maximum distance of 200 units. For each of the 8 rays, the simulation calculates the distance to the nearest:

- Wall/Boundary
- Food Pellet
- Enemy Agent

This generates an array of 24 raw distance values (8 rays × 3 metrics). These values are normalized to a scale of 0.0 to 1.0 to form the agent's **Input Tensor**.

B. Neural Inference

The 24-value input tensor is fed into the agent's neural network. Executed entirely on the GPU via PyTorch, the network processes the sensory data and instantly returns 4 output values, clamped between mathematical boundaries:

- **Thrust (0.0 to 1.0):** How much forward acceleration to apply.
- **Turn (-1.0 to 1.0):** Rotational steering (left vs. right).
- **Shoot (0.0 to 1.0):** Trigger to fire a projectile (predation mechanism).
- **Split (0.0 to 1.0):** Trigger for asexual reproduction mechanics.

C. Physics & Environmental Interactions

The neural network outputs are translated into physical movement and game logic:

- **Movement:** The agent's angle is updated by the Turn output. If Thrust exceeds a threshold (0.1), velocity is added in the direction of the angle. Velocity is capped at a maximum of 20.0 units/tick.

- **Metabolism:** Agents start with 100 energy. Every tick, energy passively decays (metabolic cost). If an agent's energy reaches 0, it dies and is removed from the active simulation.
- **Foraging:** The simulation checks for collisions between agents and food. If the Euclidean distance between an agent and a food pellet is less than their combined radii, the food is consumed. The agent instantly gains 25.0 energy, and the food pellet is marked for respawning.

3 Evaluation & Evolution (End of Generation)

After 750 ticks, the simulation halts. The timeline freezes, and the Genetic Algorithm takes over to evaluate performance and create the next generation.

Fitness Calculation

Every agent is graded based on how successfully it navigated the environment. The fitness function is designed to reward both survival duration and resource gathering:

```
Fitness = (Ticks Survived) + (Final Remaining Energy)
```

Agents that die early receive low scores. Agents that survive the full 750 ticks and hoard food receive the highest scores.

Selection (The Top 20%)

The population is sorted by Fitness Score. A strict **Selection Pressure of 0.2** is applied. The top 200 performing agents are crowned the "Elite" and are selected to reproduce. The bottom 800 agents are discarded.

Reproduction & Mutation

To repopulate back up to 1,000 agents for the next generation, the top 20% are cloned. However, nature requires variation to evolve. A mutation function introduces deliberate "errors" into the cloned neural network weights. This is where the core experiment variable lies:

- **Control Group (Static Mutation):** Every cloned child receives a flat, unchanging mutation rate ($\varepsilon = 0.05$). A randomized 5% variance is applied to their neural weights, regardless of how well their parent performed.
- **Experimental Group (Adaptive Mutation):** The mutation rate is dynamically scaled based on the parent's fitness score.

  $$\varepsilon = \text{Base\_Rate} \times (1 - (\text{Parent\_Fitness} / \text{Max\_Possible\_Fitness}))$$

  The Result: If a parent had a terrible fitness score, their children experience a massive mutation rate (high entropy), forcing rapid exploration of new strategies. If a parent had

a near-perfect fitness score, their children experience a microscopic mutation rate (low entropy), preserving the successful strategy and fine-tuning it.

4 Convergence & Nash Equilibrium

This entire process—Initialization, 750 Simulation Ticks, Fitness Evaluation, and Reproduction—loops continuously for hundreds of generations. The system tracks the **Policy Entropy** (the statistical randomness of the actions the population chooses to take).

As generations pass, natural selection weeds out bad strategies (like spinning in circles or ignoring food). The neural network weights mathematically align toward the optimal behavior profile.

Equilibrium Detection

The experiment actively monitors the variance of Policy Entropy. When the entropy variance drops below a strict threshold (0.001) and remains stable for a window of 20 consecutive generations (while maintaining a viable average fitness > 50), the system triggers a **Nash Equilibrium Detection**.

At this point, the population has converged on a singular, globally stable strategy. No individual agent can improve its fitness by unilaterally mutating or changing its behavior. The simulation has successfully mathematically proven the endpoint of evolutionary game theory.