Article

mouseColSimu: A web-based simulation model for mouse colony evolution

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou, China
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 16 December 2025; *Accepted 6 January 2026*; *Published online 15 January 2026*; *Published 1 June 2027*

Abstract
In present study a web-based simulation model for mouse colony evolution was developed. It aims to build an interactive, agent-based simulation that explores how population density, resource access, and social behaviors shape group dynamics, inspired by Calhoun's Universe experiments, provide a modular platform to test hypotheses about dominance, cooperation, stress-induced behaviors, and reproduction under crowding, using tunable parameters and stochastic rules, and enable educational visualization: demonstrate how simple, local interaction rules can yield complex, emergent colony-level patterns (growth, collapse, clustering, aggression). Full JS+HTML codes, algorithmic description and user guide were provided.

Keywords mouse colony; Calhoun's Universe; evolution; web tool.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/selforganizology /rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Research on the evolution of sociality in animals and humans seeks to explain how complex group structures arise from individual interactions and environmental constraints, combining classic ethological insights with modern network theory and evolutionary game models (Wilson, 1975; Maynard Smith & Szathmáry, 1995; Nowak, 2006). Early laboratory studies of rodent populations highlighted density-dependent stress, behavioral pathologies, and demographic collapse, famously illustrated by the Universe experiments where resource abundance failed to prevent social breakdown under crowding (Calhoun, 1973). Subsequent historical and sociological analyses examined how these rodent studies shaped public discourse on urban life and overpopulation, reframing laboratory findings within broader cultural narratives of modernity and “urban derangement” (Ramsden & Adams, 2009; Adams & Ramsden, 2024). Parallel theoretical developments established general conditions for cooperation—kin selection, direct and indirect reciprocity, network reciprocity, and group selection—offering a mathematical scaffold for understanding when prosocial traits can spread in finite populations (Nowak, 2006; West et al., 2007). Empirical and modeling work on collective motion and swarm intelligence demonstrated how local interaction rules yield emergent coordination, decision

making, and information pooling in flocks, schools, and crowds, emphasizing the dual roles of alignment and noise in stability and flexibility (Couzin, 2009; Sumpter, 2010; Krause et al., 2010). Complementing these perspectives, the framework of self-organization treats group structure as a dynamical outcome of feedbacks, constraints, and phase transitions, integrating pattern formation with network growth processes (Zhang, 2012, 2015, 2016a-b). Social network research further clarified how heterogeneity in degree, modularity, and temporal dynamics shapes the spread of cooperation and conflict, revealing that changing contact patterns can flip systems between cohesive and fragmented regimes (Pinter-Wollman et al., 2014; West et al., 2007). Synthetic volumes have emphasized the diversity of social forms, from eusocial insect colonies to human institutions, and their macroevolutionary implications for division of labor and information processing (Maynard Smith & Szathmáry, 1995; Wilson, 1975). News and perspective pieces distilled these advances for broader audiences, underscoring open questions about the robustness of cooperative strategies in fluctuating environments and the replicability of classic lab findings (Pennisi, 2005; Ramsden & Adams, 2009).

The significance of this research lies in its ability to bridge scales—from neural and hormonal stress pathways to network topology and ecological feedback—thereby informing both basic science and policy debates about urban design, welfare, and public health (Calhoun, 1973; Ramsden & Adams, 2009; Couzin, 2009). Key contributions include the identification of precise evolutionary rules that permit cooperation, which replaced purely verbal debates with testable predictions, and the demonstration that local interaction rules can generate global coordination without central control (Nowak, 2006; Sumpter, 2010; Krause et al., 2010). The self-organization literature extends these insights by formalizing how node attraction, preferential attachment, and constraint-driven optimization drive community assembly and phase shifts, offering algorithmic tools for simulating social evolution in silico (Zhang, 2012, 2015, 2016a-b). However, limitations persist: lab rodent paradigms may lack ecological validity, raising questions about extrapolation to human societies and to wild populations, while cultural reception sometimes overgeneralized or moralized findings beyond their empirical scope (Calhoun, 1973; Ramsden & Adams, 2009; Adams & Ramsden, 2024). Additionally, many cooperation models assume simplified payoff structures and static networks, whereas real systems feature multiplex ties, nonstationary environments, and cognitive biases that complicate equilibrium predictions (West et al., 2007; Pinter-Wollman et al., 2014). Moving forward, promising directions include integrating mechanistic stress physiology with adaptive network models to explain how crowding alters both behavior and contact structure, and embedding evolutionary game dynamics within empirically measured temporal networks to test causality at scale (Couzin, 2009; Pinter-Wollman et al., 2014; Nowak, 2006). Cross-disciplinary synthesis that links historical context with contemporary data science can also refine the narrative around crowding and social pathology, distinguishing when density drives dysfunction versus when institutions and spatial design buffer it (Ramsden & Adams, 2009; Adams & Ramsden, 2024). Ultimately, a pluralistic approach that combines experimental models, comparative fieldwork, and computational self-organization theory offers the best path to a predictive science of social evolution (Sumpter, 2010; Zhang, 2016a-b; West et al., 2007).

In present study, I will develop a web-based simulation model for mouse colony evolution. The simulation model aims to build an interactive, agent-based simulation that explores how population density, resource access, and social behaviors shape group dynamics, inspired by Calhoun's Universe experiments, provide a modular platform to test hypotheses about dominance, cooperation, stress-induced behaviors, and reproduction under crowding, using tunable parameters and stochastic rules, and enable educational visualization: demonstrate how simple, local interaction rules can yield complex, emergent colony-level patterns.

2 Overview

Objective

- Build an interactive, agent-based simulation that explores how population density, resource access, and social behaviors shape group dynamics, inspired by Calhoun's Universe experiments.
- Provide a modular platform to test hypotheses about dominance, cooperation, stress-induced behaviors, and reproduction under crowding, using tunable parameters and stochastic rules.
- Enable educational visualization: demonstrate how simple, local interaction rules can yield complex, emergent colony-level patterns (growth, collapse, clustering, aggression).

Overview

- Core model
- Agents (mice) live on a 2D canvas with attributes: position, age, energy, gender, behavior, and reproduction cooldown.
- Time advances in discrete steps; each step updates aging, energy, behavior transitions, movement, interactions, reproduction, and rendering.
- Behaviors include normal, dominant, rebel, sleepwalker, and pansexual; transitions depend on local density, energy, age, and stochastic events.
- Dynamics
- Local crowding raises dominance/rebel states and aggression; low energy and age increase sleepwalking likelihood; rare mutations introduce pansexual behavior affecting mating compatibility.
- Movement blends random wandering with repulsion from crowded centroids, producing clusters and dispersal patterns.
- Reproduction is proximity-, energy-, and cooldown-limited; offspring inherit traits with mutation probabilities.
- Computation
- Baseline neighbor search is O(N^2) for clarity; optional spatial hashing reduces to near O(N).
- Rendering maps age/behavior to size and color; info panels summarize population and behavior distributions.
- Parameters (e.g., overcrowding threshold, move distance, reproduction energy) are exposed for experimentation and sensitivity analysis.
- Extensibility
- Hooks for food patches, age-gated breeding, seeded RNG for reproducibility, and UI sliders.
- Clear separation between model state update and rendering for performance and testing.

Significance

- **Scientific/educational value**
- Demonstrates self-organization and emergent social phenomena from local rules, connecting to classic and contemporary theories of social evolution and collective behavior.
- Offers a sandbox to interrogate density-dependent effects, aggression, and reproductive bottlenecks—illustrating when populations boom, stabilize, or collapse.
- Bridges qualitative narratives (e.g., crowding and social pathology) with quantitative, experimentable models that can be parameter-swept and replicated.
- **Methodological impact**
- Encodes testable mechanisms as transparent algorithms, making assumptions explicit (e.g., energy budgets, local interactions, probabilistic transitions).
- Encourages rigorous exploration: alter a parameter, observe macroscopic outcomes, record trajectories with seeded randomness, and compare scenarios.

- Provides a lightweight template for agent-based modeling in the browser—easy to teach, extend, and share.
- **Practical relevance**
- Helps learners and researchers visualize how design choices (space, resources, thresholds) influence social structure and stability.
- Serves as a starting point for more realistic models (physiology, adaptive networks, resource dynamics) and for UI-driven "experiments" that communicate complex systems intuitively.

3 Algorithmic Description

1. State representation

- Agents (mice)
  - Continuous 2D position: $(x, y) \in [0, W] \times [0, H]$
  - Attributes:
    - gender $\in$ {male, female}
    - behavior $\in$ {normal, dominant, rebel, sleepwalker, pansexual}
    - age $\in \mathbb{N}$ (steps)
    - energy $\in$ [0, ENERGY_MAX]
    - alive $\in$ {true, false}
    - reproCooldown $\in \mathbb{N}0$
    - sleepwalkingPhase $\in \mathbb{N}0$
- Global parameters (constants)
  - canvas size: W = 800, H = 600
  - NUM_INITIAL_MICE = 100
  - MAX_AGE = 80
  - BREEDING_AGE_MIN, BREEDING_AGE_MAX (currently defined but not enforced in code)
  - ENERGY_MAX = 100, ENERGY_LOSS_PER_STEP = 1, ENERGY_GAIN_FROM_FOOD (placeholder)
  - MOVE_DISTANCE = 3
  - REPRODUCTION_ENERGY_THRESHOLD = 50, REPRODUCTION_ENERGY_COST = 20
  - NEIGHBORHOOD_RADIUS = 20
  - OVERCROWDING_THRESHOLD = 15
  - TIME_STEP_DELAY = 50 ms

2. **Simulation loop per time step t**

For each alive mouse i:

- Aging and survival
  - age_i ← age_i + 1
  - energy_i ← energy_i − ENERGY_LOSS_PER_STEP
  - If energy_i ≤ 0 or age_i > MAX_AGE ⇒ alive_i ← false
  - reproCooldown_i ← max(0, reproCooldown_i − 1)
- Behavior update (behavioral state machine)
  - Count neighbors within NEIGHBORHOOD_RADIUS:

- neighbors_i = { j ≠ i, alive_j = true, ||pos_j − pos_i||^2 < R^2 }, R = NEIGHBORHOOD_RADIUS
- n_i = |neighbors_i|

o Male dominance/rebel formation:

- If gender_i = male:
- If n_i > OVERCROWDING_THRESHOLD and age_i > 20 and energy_i > 0.6·ENERGY_MAX:
- behavior_i ← dominant
- energy_i ← min(ENERGY_MAX, energy_i + 20)
- Else if n_i > OVERCROWDING_THRESHOLD/2 and age_i > 15:
- behavior_i ← rebel
- Else:
- behavior_i ← normal

o Sleepwalker onset:

- If behavior_i ≠ sleepwalker and energy_i < 0.2·ENERGY_MAX and age_i > 30 and Bernoulli(0.005):
- behavior_i ← sleepwalker
- sleepwalkingPhase_i ← UniformInt[20, 49]
- If behavior_i = sleepwalker:
- sleepwalkingPhase_i ← sleepwalkingPhase_i − 1
- If sleepwalkingPhase_i ≤ 0 ⇒ behavior_i ← normal

o Pansexual mutation:

- If behavior_i ≠ pansexual and Bernoulli(0.001) ⇒ behavior_i ← pansexual

• Movement

o If behavior_i = sleepwalker:

- pos_i ← pos_i + 0.5·MOVE_DISTANCE·(ξx, ξy), where ξx, ξy ∼ Uniform[-0.5, 0.5]

o Else:

- Compute centroid of neighbors C_i = (mean x, mean y) over neighbors_i (if any)
- If n_i > 0: move away from centroid
- v = pos_i − C_i
- pos_i ← pos_i + 0.5·(v_x', v_y'), with fallback (dx || 1), (dy || 1) to avoid zero vector
- Else: random wander
- pos_i ← pos_i + MOVE_DISTANCE·(ξx, ξy), ξx, ξy ∼ Uniform[-0.5, 0.5]

o Clamp position to domain bounds [2, W−2] × [2, H−2]

• Interaction

o Aggression by dominant/rebel males:

- If behavior_i ∈ {dominant, rebel}:
- For neighbors j within distance ≤ 10:
- If gender_j = male and behavior≠j behavior_i and energy_i > energy_j + 20:
- With probability 0.3 ⇒ kill j: alive_j← false; energy_i ← min(ENERGY_MAX, energy_i + 15)

- Crowding infanticide by females:
  - If gender_i = female:
    - Y_i = { young neighbors with age < 10 and alive }
    - If |Y_i| > 5:
      - For each y ∈ Y_i: with prob 0.2 ⇒ alive_y ← false
- Reproduction
  - If energy_i > REPRODUCTION_ENERGY_THRESHOLD and reproCooldown_i = 0:
    - Candidate partners P_i = { j | j≠ i, alive_j, ||pos_j − pos_i|| ≤ 15, reproCooldown_j = 0 }
    - Iterate partners in arbitrary order:
      - If canMate(i, j) = true and both energies > threshold:
        - energy_i ← energy_i − REPRODUCTION_ENERGY_COST
        - energy_j ← energy_j − REPRODUCTION_ENERGY_COST
        - reproCooldown_i ← 20; reproCooldown_j ← 20
        - Create offspring k near midpoint:
          - pos_k = (pos_i + pos_j)/2 + Uniform noise in [−5, 5] per axis
          - gender_k ~ Bernoulli(0.5) male/female
          - behavior_k = normal with prob 0.98; pansexual with prob 0.02
          - age_k = 0; energy_k ∈ [0.7·ENERGY_MAX, ENERGY_MAX]
        - Add k to population; break
- Rendering
  - Draw each mouse as a circle:
    - radius: 3 if age < 15; 6 if age > 50; else 4
    - fill color by behavior and gender
    - overlay white sex glyph (♂/♀) for alive mice
    - dead mice drawn gray for a few steps (visual persistence)
- Info panel
  - Counts: totals, gender splits, behavior types, age brackets

3. Computational considerations

- Complexity per step:
  - The current implementation uses O(N^2) neighbor queries (countNeighbors/getNeighbors scan all mice). With N ~ 100–1000 this is fine; beyond that, it will slow.
- Optimization strategies (if scaling up):
  - Spatial hashing or uniform grid binning for neighbor searches:
    - Bucket size ≈ NEIGHBORHOOD_RADIUS
    - Reduces queries to O(N) bucketing + O(k) local checks per agent
  - Separate loops for state update and interactions to avoid side-effects during iteration
  - Use typed arrays for positions/energies if very large N

4. Behavioral logic details

- Dominance/rebel thresholds:
  - Driven by local density and age/energy

- Dominant → gains energy upon “promotion” and upon successful aggression
- Sleepwalker:
- Triggered by low energy + age; transient state with reduced, random movement
- Pansexual:
- Two entry paths:
  - Spontaneous rare mutation in adults (p = 0.001 per step)
  - Offspring mutation (p = 0.02)
- Mating rule: pansexual agents can mate with anyone nearby (ignores gender pairing)
- Mating compatibility function canMate(i, j):
- If either is pansexual ⇒ true
- Else if genders differ:
  - If either is dominant and the other is female ⇒ true
  - Else if neither is dominant ⇒ true
- Else ⇒ false
- Note: BREEDING_AGE_MIN/MAX are currently not enforced; you can add age gating for realism.

Key mathematical procedures

- Distance checks:
- Use squared distance d2 = (x2 − x1)^2 + (y2 − y1)^2 to avoid sqrt; compare with R^2
- Centroid-based repulsion:
- For neighbors Ni = {p1, …, pk}, centroid C = (Σx/ k, Σy/ k)
- Update pos by pos ← pos + α(pos − C), α = 0.5
- Energy and survival:
- energy_t+1 = max(0, energy_t − c); if 0 ⇒ death
- Probabilistic transitions:
- Bernoulli(p) events for sleepwalker onset, pansexual mutation, aggression kill, female infanticide
- Offspring placement:
- midpoint + ε, with εx, εy ∼ Uniform[−5, 5]

Demo JS snippets

A) Add breeding age gates

```
js
function isBreedingAge(m) {
  return m.age >= BREEDING_AGE_MIN && m.age <= BREEDING_AGE_MAX;
}

if (this.energy > REPRODUCTION_ENERGY_THRESHOLD && this.reproCooldown === 0 && isBreedingAge(this)) {
  let potentialPartners = mice.filter(m =>
    m.alive &&
    m !== this &&
    isBreedingAge(m) &&
    dist2(this.x, this.y, m.x, m.y) < 15 * 15 &&
    m.reproCooldown === 0
  );
```

```
    // ... rest as before
}
```

B) Spatial hashing to accelerate neighbor queries

```
js
class SpatialHash {
    constructor(cellSize, width, height) {
        this.cellSize = cellSize;
        this.cols = Math.ceil(width / cellSize);
        this.rows = Math.ceil(height / cellSize);
        this.buckets = new Map();
    }
    key(cx, cy) { return `${cx},${cy}`; }
    cell(x, y) { return [Math.floor(x / this.cellSize), Math.floor(y / this.cellSize)]; }
    clear() { this.buckets.clear(); }
    insert(mouse) {
        const [cx, cy] = this.cell(mouse.x, mouse.y);
        const k = this.key(cx, cy);
        if (!this.buckets.has(k)) this.buckets.set(k, []);
        this.buckets.get(k).push(mouse);
    }
    queryCircle(x, y, r) {
        const [cx, cy] = this.cell(x, y);
        const rad = Math.ceil(r / this.cellSize);
        const results = [];
        for (let ix = cx - rad; ix <= cx + rad; ix++) {
            for (let iy = cy - rad; iy <= cy + rad; iy++) {
                const k = this.key(ix, iy);
                const bucket = this.buckets.get(k);
                if (!bucket) continue;
                for (const m of bucket) {
                    if (dist2(x, y, m.x, m.y) < r * r) results.push(m);
                }
            }
        }
        return results;
    }
}

// In Simulation:
const grid = new SpatialHash(NEIGHBORHOOD_RADIUS, CANVAS_WIDTH, CANVAS_HEIGHT);

// Each update step:
grid.clear();
for (const m of this.mice) if (m.alive) grid.insert(m);
```

```
// Replace getNeighbors/countNeighbors:
Mouse.prototype.getNeighbors = function(mice, grid) {
  return grid.queryCircle(this.x, this.y, NEIGHBORHOOD_RADIUS).filter(m => m !== this && m.alive);
};
Mouse.prototype.countNeighbors = function(mice, grid) {
  return this.getNeighbors(mice, grid).length;
};

// Pass grid to step():
mouse.step(this.mice, grid);
```

C) Tunable parameters UI (simple sliders)

```
html
<div style="padding:8px;border-top:1px solid #eee">
  <label>Move distance <input id="moveSlider" type="range" min="1" max="6" value="3"></label>
  <label>Overcrowding <input id="crowdSlider" type="range" min="5" max="30" value="15"></label>
</div>
<script>
  const moveSlider = document.getElementById('moveSlider');
  const crowdSlider = document.getElementById('crowdSlider');
  moveSlider.addEventListener('input', () => window.MOVE_DISTANCE = Number(moveSlider.value));
  crowdSlider.addEventListener('input', () => window.OVERCROWDING_THRESHOLD = Number(crowdSlider.value));
</script>
```

D) Add simple food patches (energy gain)

```
js
const FOOD_GAIN = 8;
const FOOD_CELLS = 10;
const FOOD_RADIUS = 12;
const foodPatches = Array.from({length: FOOD_CELLS}, () => ({
  x: Math.random() * CANVAS_WIDTH,
  y: Math.random() * CANVAS_HEIGHT
}));

function drawFood(ctx) {
  ctx.fillStyle = '#44bb44';
  for (const f of foodPatches) {
    ctx.beginPath();
    ctx.arc(f.x, f.y, FOOD_RADIUS, 0, 2 * Math.PI);
    ctx.fill();
  }
}

// In Mouse.step after move():
```

```
for (const f of foodPatches) {
  if (dist2(this.x, this.y, f.x, f.y) < FOOD_RADIUS * FOOD_RADIUS) {
    this.energy = Math.min(ENERGY_MAX, this.energy + FOOD_GAIN);
  }
}

// In Simulation.draw before mice:
drawFood(this.ctx);
```

4 User Guide

- **Goal**
  - Observe emergent dynamics inspired by Universe 25 themes: population growth, dominance, aggression under crowding, and behavioral shifts.
- **Controls and pacing**
  - Simulation autostarts and runs at TIME_STEP_DELAY ms per tick.
  - To pause: wrap animate() loop with a running flag; to change speed, adjust TIME_STEP_DELAY.
  - To change initial conditions: set NUM_INITIAL_MICE and constants in the script header.
- **Reading the visualization**
  - Colors:
    - Dominant male: blue
    - Rebel male: red
    - Normal male: green
    - Normal female: orange
    - Sleepwalker: yellow
    - Pansexual: purple
    - Dead: gray (fades after a few steps)
  - Size:
    - Small = young (<15)
    - Large = old (>50)
  - Symbol:
    - ♂ male, ♀ female
  - Info panel:
    - Live counts per category update each tick.
- **Typical scenarios to try**
  - Increase overcrowding threshold to reduce dominance/rebel formation; watch population growth accelerate.
  - Lower REPRODUCTION_ENERGY_THRESHOLD to boost birth rates, then observe energy scarcity and mortality.
  - Add spatial hashing if you scale population > 1000 to keep frame rate smooth.
  - Introduce food patches to create aggregation and competition hotspots.
- **Performance tips**
  - Reduce rendering work: only draw alive mice and a small fade window for dead ones.
  - Batch state updates, then a separate pass for rendering.

- Use requestAnimationFrame for smoother visuals if you don't need a fixed step delay; combine with an accumulator for fixed-step logic.

- **Reproducibility**
  - For deterministic replays, seed Math.random() with a PRNG (e.g., mulberry32 or xorshift) and replace Math.random() calls.

Example: seeded RNG

```
js
function mulberry32(a) {
  return function() {
    let t = a += 0x6D2B79F5;
    t = Math.imul(t ^ (t >>> 15), t | 1);
    t ^= t + Math.imul(t ^ (t >>> 7), t | 61);
    return ((t ^ (t >>> 14)) >>> 0) / 4294967296;
  }
}
// Usage:
const rand = mulberry32(123456789);
function randUniform() { return rand(); }
// Replace Math.random() with randUniform()
```

5 Validation Checklist

- Agent lifecycle
  - Ages increment; deaths at zero energy or age > MAX_AGE
- Behavior transitions
  - Dominance/rebel depend on local density and age/energy
  - Sleepwalker rare onset under stress; temporary
  - Pansexual rare spontaneous or inherited
- Movement
  - Sleepwalker: slow random
  - Others: repel from local centroid or wander
- Interaction
  - Male aggression within 10px; energy and kill probability modeled
  - Female stress infanticide under local overcrowding
- Reproduction
  - Proximity-limited, energy-limited, cooldown-limited, with compatibility rules

6 Complete Codes

The following are the HTML+JavaScript codes (http://www.iaees.org/publications/journals/selforganizology/articles/2027-14(1-2)/mouseColSimu.htm; Fig. 1):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Universe 25 Mouse Colony Simulation</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      display: flex;
      flex-direction: row;
      background: #f0f0f0;
      height: 100vh;
    }
    #canvas {
      border: 1px solid #ccc;
      background: #fff;
      flex-grow: 1;
      display: block;
    }
    #infoPanel {
      width: 260px;
      padding: 15px;
      background: #fff;
      border-left: 1px solid #ccc;
      overflow-y: auto;
    }
    h2 {
      margin-top: 0;
      font-size: 1.2em;
    }
    .stat {
      margin-bottom: 10px;
    }
    .legend {
      margin-top: 15px;
      font-size: 0.9em;
      line-height: 1.4;
    }
    .colorBox {
      display: inline-block;
      width: 14px;
      height: 14px;
      margin-right: 5px;
      vertical-align: middle;
    }
  </style>
</head>
```

```
<body>
  <canvas id="canvas" width="800" height="600"></canvas>
  <div id="infoPanel">
    <h2>Mouse Colony Stats</h2>
    <div class="stat" id="populationTotal">Total Population: 0</div>
    <div class="stat" id="populationMale">Males: 0</div>
    <div class="stat" id="populationFemale">Females: 0</div>
    <div class="stat" id="populationNormal">Normal Mice: 0</div>
    <div class="stat" id="populationDominant">Dominant Males: 0</div>
    <div class="stat" id="populationRebel">Rebel Males: 0</div>
    <div class="stat" id="populationSleepwalker">Sleepwalkers: 0</div>
    <div class="stat" id="populationPansexual">Pansexual Mice: 0</div>
    <div class="stat" id="populationYoung">Young (<15 steps): 0</div>
    <div class="stat" id="populationOld">Old (>50 steps): 0</div>
    <div class="legend">
      <div><span class="colorBox" style="background:#0000cc;"></span>Dominant Male</div>
      <div><span class="colorBox" style="background:#cc0000;"></span>Rebel Male</div>
      <div><span class="colorBox" style="background:#007700;"></span>Normal Male</div>
      <div><span class="colorBox" style="background:#cc7700;"></span>Normal Female</div>
      <div><span class="colorBox" style="background:#cccc00;"></span>Sleepwalker</div>
      <div><span class="colorBox" style="background:#770099;"></span>Pansexual</div>
      <div><span class="colorBox" style="background:#999999;"></span>Dead (gray)</div>
    </div>
  </div>

  <script>
    // --- Configuration Parameters ---
    const CANVAS_WIDTH = 800;
    const CANVAS_HEIGHT = 600;
    const NUM_INITIAL_MICE = 100;
    const MAX_AGE = 80; // mouse dies after this age in steps
    const BREEDING_AGE_MIN = 10;
    const BREEDING_AGE_MAX = 60;
    const ENERGY_MAX = 100;
    const ENERGY_LOSS_PER_STEP = 1;
    const ENERGY_GAIN_FROM_FOOD = 20; // placeholder for food model
    const MOVE_DISTANCE = 3;
    const REPRODUCTION_ENERGY_THRESHOLD = 50;
    const REPRODUCTION_ENERGY_COST = 20;
    const OVERCROWDING_THRESHOLD = 15; // number of mice in neighborhood for crowding
    const NEIGHBORHOOD_RADIUS = 20;
    const TIME_STEP_DELAY = 50; // ms between steps

    // --- Behavior States ---
    const BEHAVIOR = {
```

```
    NORMAL: 'normal',              // Regular mouse
    DOMINANT: 'dominant',          // Dominant male
    REBEL: 'rebel',                // Aggressive non-dominant male
    SLEEPWALKER: 'sleepwalker', // State leading to random behavior
    PANSEXUAL: 'pansexual'      // No strict gender preference
};

// --- Gender Constants ---
const GENDER = {
    MALE: 'male',
    FEMALE: 'female'
};

// --- Color Map for Rendering ---
const COLOR_MAP = {
    [BEHAVIOR.DOMINANT]: '#0000cc',
    [BEHAVIOR.REBEL]: '#cc0000',
    [BEHAVIOR.NORMAL]: {
        [GENDER.MALE]: '#007700',
        [GENDER.FEMALE]: '#cc7700',
    },
    [BEHAVIOR.SLEEPWALKER]: '#cccc00',
    [BEHAVIOR.PANSEXUAL]: '#770099',
    dead: '#999999',
};

// Utility function: distance squared between two points
function dist2(x1, y1, x2, y2) {
    let dx = x2 - x1;
    let dy = y2 - y1;
    return dx * dx + dy * dy;
}

class Mouse {
    constructor(x, y, gender, behavior) {
        this.x = x;
        this.y = y;
        this.gender = gender;                    // 'male' or 'female'
        this.behavior = behavior;                // One of BEHAVIOR
        this.age = 0;                            // Age in steps
        this.energy = Math.floor(ENERGY_MAX * 0.7 + Math.random() * ENERGY_MAX * 0.3);
        this.alive = true;
        this.sleepwalkingPhase = 0;              // counter for sleepwalker behavior cycles
        this.reproCooldown = 0;                  // time between reproductions
    }
```

```
isOld() {
    return this.age > 50;
}

isYoung() {
    return this.age < 15;
}

step(mice, grid) {
    if (!this.alive) return;

    this.age++;
    this.energy -= ENERGY_LOSS_PER_STEP;

    if (this.energy <= 0 || this.age > MAX_AGE) {
        this.die();
        return;
    }

    this.reproCooldown = Math.max(0, this.reproCooldown - 1);

    // 1. Behavior evolution based on environment and age
    this.updateBehavior(mice);

    // 2. Move
    this.move(mice);

    // 3. Interaction: attacking weaker mice, crowding effects
    this.interact(mice);

    // 4. Reproduction attempt
    if (this.energy > REPRODUCTION_ENERGY_THRESHOLD && this.reproCooldown === 0) {
        this.tryReproduce(mice);
    }
}

updateBehavior(mice) {
    // Dominant male formation: strong, older males in dense areas
    if (this.gender === GENDER.MALE && this.behavior !== BEHAVIOR.DOMINANT) {
        let neighbors = this.countNeighbors(mice);
        if (neighbors > OVERCROWDING_THRESHOLD && this.age > 20 && this.energy > ENERGY_MAX * 0.6) {
            this.behavior = BEHAVIOR.DOMINANT;
            // Becoming dominant refreshes energy a bit
            this.energy = Math.min(ENERGY_MAX, this.energy + 20);
```

```
            }
            // Else become rebel (aggressive non dominant)
            else if (neighbors > OVERCROWDING_THRESHOLD / 2 && this.age > 15 && this.behavior !== BEHAVIOR.DOMINANT) {
                this.behavior = BEHAVIOR.REBEL;
            }
            else {
                this.behavior = BEHAVIOR.NORMAL;
            }
        }

        // Sleepwalker behavior randomly initiates based on energy and age
        if (this.behavior !== BEHAVIOR.SLEEPWALKER && this.energy < ENERGY_MAX * 0.2 && this.age > 30) {
            if (Math.random() < 0.005) { // rare chance to become sleepwalker
                this.behavior = BEHAVIOR.SLEEPWALKER;
                this.sleepwalkingPhase = 20 + Math.floor(Math.random() * 30);
            }
        }

        if (this.behavior === BEHAVIOR.SLEEPWALKER) {
            this.sleepwalkingPhase--;
            if (this.sleepwalkingPhase <= 0) {
                // Return to normal after sleepwalking phase
                this.behavior = BEHAVIOR.NORMAL;
            }
        }

        // Pansexual: rare mutation or behavioral anomaly, unaffected by gender in reproduction
        if (this.behavior !== BEHAVIOR.PANSEXUAL) {
            if (Math.random() < 0.001) {
                this.behavior = BEHAVIOR.PANSEXUAL;
            }
        }
    }

    countNeighbors(mice) {
        let count = 0;
        for (let m of mice) {
            if (m !== this && m.alive) {
                let d = dist2(this.x, this.y, m.x, m.y);
                if (d < NEIGHBORHOOD_RADIUS * NEIGHBORHOOD_RADIUS) {
                    count++;
                }
            }
        }
```

```
    return count;
  }

  move(mice) {
    if (this.behavior === BEHAVIOR.SLEEPWALKER) {
      // Random slow move
      this.x += (Math.random() - 0.5) * MOVE_DISTANCE * 0.5;
      this.y += (Math.random() - 0.5) * MOVE_DISTANCE * 0.5;
      this.clampPosition();
      return;
    }

    // Normal/dominant/rebel/pansexual movement is toward less crowded area or random wander

    let neighbors = this.getNeighbors(mice);
    if (neighbors.length > 0) {
      // move away from crowded center
      let centerX = 0;
      let centerY = 0;
      for (let n of neighbors) {
        centerX += n.x;
        centerY += n.y;
      }
      centerX /= neighbors.length;
      centerY /= neighbors.length;
      let dx = this.x - centerX;
      let dy = this.y - centerY;
      // Small vector pushing away from neighbors center
      this.x += (dx || 1) * 0.5;
      this.y += (dy || 1) * 0.5;
    } else {
      // Wander randomly
      this.x += (Math.random() - 0.5) * MOVE_DISTANCE;
      this.y += (Math.random() - 0.5) * MOVE_DISTANCE;
    }
    this.clampPosition();
  }

  getNeighbors(mice) {
    return mice.filter(m => m !== this && m.alive && dist2(this.x, this.y, m.x, m.y) < NEIGHBORHOOD_RADIUS *
NEIGHBORHOOD_RADIUS);
  }

  clampPosition() {
    this.x = Math.min(CANVAS_WIDTH - 2, Math.max(2, this.x));
```

```
        this.y = Math.min(CANVAS_HEIGHT - 2, Math.max(2, this.y));
    }

    interact(mice) {
        // Aggressive males attack weaker males nearby
        if (this.behavior === BEHAVIOR.DOMINANT || this.behavior === BEHAVIOR.REBEL) {
            for (let m of mice) {
                if (!m.alive) continue;
                if (m === this) continue;
                if (m.gender === GENDER.MALE && this.behavior !== m.behavior && dist2(this.x, this.y, m.x, m.y) < 10 * 10)
{
                    // Attack if weaker (less energy)
                    if (this.energy > m.energy + 20) {
                        // Kill target with some probability
                        if (Math.random() < 0.3) {
                            m.die();
                            this.energy = Math.min(ENERGY_MAX, this.energy + 15);
                        }
                    }
                }
            }
        }

        // Crowding effect: females may attack young mice when crowding is too high (stress)
        if (this.gender === GENDER.FEMALE) {
            let localMice = this.getNeighbors(mice).filter(m => m.age < 10 && m.alive);
            if (localMice.length > 5) {
                // higher chance to kill young mice
                for (let young of localMice) {
                    if (Math.random() < 0.2) {
                        young.die();
                    }
                }
            }
        }
    }

    tryReproduce(mice) {
        // Only breed between compatible partners nearby

        // Pansexual mice breed with any nearby mouse of opposite gender or pansexual
        // Normal mice follow gender division, dominant male prefer breeding first

        let potentialPartners = mice.filter(m => m.alive && m !== this && dist2(this.x, this.y, m.x, m.y) < 15 * 15 &&
m.reproCooldown === 0);
```

```
    for (let partner of potentialPartners) {
      if (!this.canMateWith(partner)) continue;

      // Both lose energy and create offspring
      if (this.energy > REPRODUCTION_ENERGY_THRESHOLD && partner.energy > REPRODUCTION_ENERGY_THRESHOLD) {
        this.energy -= REPRODUCTION_ENERGY_COST;
        partner.energy -= REPRODUCTION_ENERGY_COST;
        this.reproCooldown = 20;
        partner.reproCooldown = 20;
        mice.push(this.createOffspringWith(partner));
        break; // reproduce once per step max
      }
    }
  }

  canMateWith(partner) {
    if (!partner.alive) return false;

    // Pansexual mate with anyone except self
    if (this.behavior === BEHAVIOR.PANSEXUAL || partner.behavior === BEHAVIOR.PANSEXUAL) return true;

    // Normal gender-based mating
    if (this.gender !== partner.gender) {
      // Dominant males prefer females, rebels less selective
      if (this.behavior === BEHAVIOR.DOMINANT && partner.gender === GENDER.FEMALE) return true;
      if (partner.behavior === BEHAVIOR.DOMINANT && this.gender === GENDER.FEMALE) return true;

      if (this.behavior !== BEHAVIOR.DOMINANT && partner.behavior !== BEHAVIOR.DOMINANT) return true;
    }
    return false;
  }

  createOffspringWith(partner) {
    // Offspring is near parents with some variations
    let ox = (this.x + partner.x) / 2 + (Math.random() - 0.5) * 10;
    let oy = (this.y + partner.y) / 2 + (Math.random() - 0.5) * 10;
    ox = Math.min(CANVAS_WIDTH - 5, Math.max(5, ox));
    oy = Math.min(CANVAS_HEIGHT - 5, Math.max(5, oy));

    // Gender with roughly equal probability
    let gender = Math.random() < 0.5 ? GENDER.MALE : GENDER.FEMALE;

    // Behavior is usually normal, but mutation may occur
```

```
    let behavior = BEHAVIOR.NORMAL;

    if (Math.random() < 0.02) {
      behavior = BEHAVIOR.PANSEXUAL; // mutation rare
    }

    return new Mouse(ox, oy, gender, behavior);
  }

  die() {
    this.alive = false;
  }

  draw(ctx) {
    if (!this.alive) {
      ctx.fillStyle = COLOR_MAP.dead;
      ctx.beginPath();
      ctx.arc(this.x, this.y, 4, 0, 2 * Math.PI);
      ctx.fill();
      return;
    }

    let color = COLOR_MAP[this.behavior];
    if (typeof color === 'object') {
      color = color[this.gender];
    }
    ctx.fillStyle = color || '#000000';

    // Draw circle sized by age bracket
    let radius = 4;
    if (this.isYoung()) {
      radius = 3;
    } else if (this.isOld()) {
      radius = 6;
    }

    ctx.beginPath();
    ctx.arc(this.x, this.y, radius, 0, 2 * Math.PI);
    ctx.fill();

    // Draw a symbol for males or females
    ctx.fillStyle = '#ffffff';
    ctx.font = '8px Arial';
    ctx.textAlign = 'center';
    ctx.textBaseline = 'middle';
```

```
        ctx.fillText(this.gender === GENDER.MALE ? '♂' : '♀', this.x, this.y);
    }
}

class Simulation {
    constructor() {
        this.canvas = document.getElementById('canvas');
        this.ctx = this.canvas.getContext('2d');
        this.mice = [];

        // Initialize population
        for (let i = 0; i < NUM_INITIAL_MICE; i++) {
            let x = Math.random() * CANVAS_WIDTH;
            let y = Math.random() * CANVAS_HEIGHT;
            let gender = Math.random() < 0.5 ? GENDER.MALE : GENDER.FEMALE;
            let behavior = BEHAVIOR.NORMAL;
            this.mice.push(new Mouse(x, y, gender, behavior));
        }

        this.stepCount = 0;
    }

    update() {
        this.stepCount++;

        for (let mouse of this.mice) {
            mouse.step(this.mice);
        }

        // Optionally remove dead mice after some steps
        this.mice = this.mice.filter(m => m.alive || (this.stepCount - m.age) < 5); // keep dead mice briefly for visuals
    }

    draw() {
        this.ctx.clearRect(0, 0, CANVAS_WIDTH, CANVAS_HEIGHT);

        for (let mouse of this.mice) {
            mouse.draw(this.ctx);
        }
    }

    updateInfoPanel() {
        const total = this.mice.filter(m => m.alive).length;
        const males = this.mice.filter(m => m.alive && m.gender === GENDER.MALE).length;
        const females = this.mice.filter(m => m.alive && m.gender === GENDER.FEMALE).length;
```

```
            const normal = this.mice.filter(m => m.alive && m.behavior === BEHAVIOR.NORMAL).length;
            const dominant = this.mice.filter(m => m.alive && m.behavior === BEHAVIOR.DOMINANT).length;
            const rebel = this.mice.filter(m => m.alive && m.behavior === BEHAVIOR.REBEL).length;
            const sleepwalker = this.mice.filter(m => m.alive && m.behavior === BEHAVIOR.SLEEPWALKER).length;
            const pansexual = this.mice.filter(m => m.alive && m.behavior === BEHAVIOR.PANSEXUAL).length;
            const young = this.mice.filter(m => m.alive && m.isYoung()).length;
            const old = this.mice.filter(m => m.alive && m.isOld()).length;

            document.getElementById('populationTotal').textContent = `Total Population: ${total}`;
            document.getElementById('populationMale').textContent = `Males: ${males}`;
            document.getElementById('populationFemale').textContent = `Females: ${females}`;
            document.getElementById('populationNormal').textContent = `Normal Mice: ${normal}`;
            document.getElementById('populationDominant').textContent = `Dominant Males: ${dominant}`;
            document.getElementById('populationRebel').textContent = `Rebel Males: ${rebel}`;
            document.getElementById('populationSleepwalker').textContent = `Sleepwalkers: ${sleepwalker}`;
            document.getElementById('populationPansexual').textContent = `Pansexual Mice: ${pansexual}`;
            document.getElementById('populationYoung').textContent = `Young (<15 steps): ${young}`;
            document.getElementById('populationOld').textContent = `Old (>50 steps): ${old}`;
        }

        run() {
            this.update();
            this.draw();
            this.updateInfoPanel();
        }
    }

    // Initialize and run simulation
    const sim = new Simulation();

    function animate() {
        sim.run();
        setTimeout(animate, TIME_STEP_DELAY);
    }

    animate();
  </script>
</body>
</html>
```

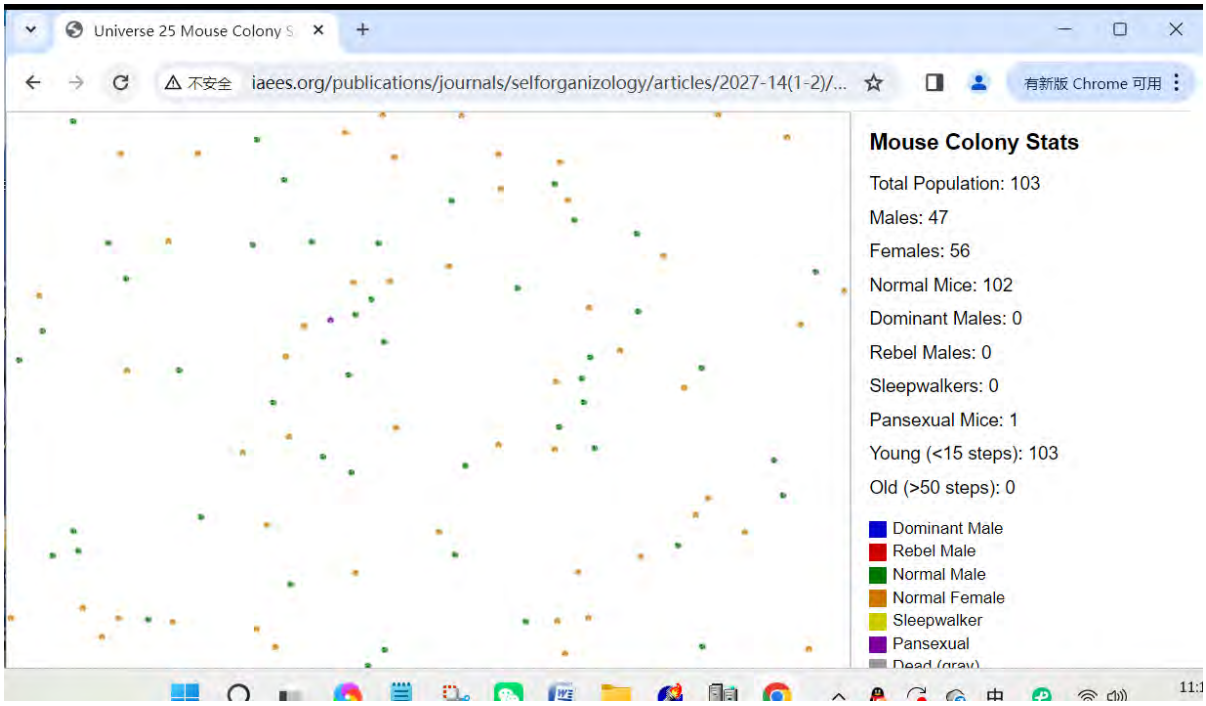

Fig. 1 mouseColSimu.

7 Discussion

This is a stylized, agent-based toy model. It captures some qualitative behaviors but is not a calibrated biological model. Neighbor interactions are stochastic; runs are non-deterministic. BREEDING_AGE_MIN/MAX are defined but not enforced by default; enable with the snippet above for age realism. The model can be further improved in the future.

References

Adams J, Ramsden E. 2024. Rat City: Overcrowding and Urban Derangement in the Rodent Universes of John B. Calhoun. Melville House. https://www.amazon.com/Rat-City-Overcrowding-Derangement-Universes/dp/1685890997

Calhoun JB. 1973. Death squared: the explosive growth and demise of a mouse population. Proceedings of the Royal Society of Medicine, 66(1 Pt 2): 80–88. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1644264/

Couzin ID. 2009. Collective cognition in animal groups. Trends in Cognitive Sciences, 13(1):36–43. https://www.cell.com/trends/cognitive-sciences/fulltext/S1364-6613(08)00317-9

Krause J, Ruxton GD, Krause S. 2010. Swarm intelligence in animals and humans. Trends in Ecology & Evolution, 25(1): 28–34. https://www.cell.com/trends/ecology-evolution/fulltext/S0169-5347(09)00263-5

Maynard Smith J, Szathmáry E. 1995. The Major Transitions in Evolution. Oxford University Press. https://global.oup.com/academic/product/the-major-transitions-in-evolution-9780198502944

Nowak MA. 2006. Five rules for the evolution of cooperation. Science, 314(5805): 1560–1563. https://www.science.org/doi/10.1126/science.1133755

Pennisi E. 2005. How did cooperative behavior evolve? Science, 309(5731): 93.

https://www.science.org/doi/10.1126/science.309.5731.93

Pinter-Wollman N, et al. 2014. The dynamics of animal social networks: analytical, conceptual, and theoretical advances. Behavioral Ecology, 25(2):242–255. https://academic.oup.com/beheco/article/25/2/242/287719

Ramsden E, Adams J. 2009. Escaping the Laboratory: The Rodent Experiments of John B. Calhoun and Their Cultural Influence. Journal of Social History, 42: 761–792. Abstract/info:. https://www.sciencehistory.org/stories/magazine/mouse-heaven-or-mouse-hell/

Sumpter DJT. 2010. Collective Animal Behavior. Princeton University Press, USA. https://press.princeton.edu/books/paperback/9780691148434/collective-animal-behavior?srsltid=AfmBOoqjnvopRhwYx691gBe5ZSc1ANugC_a3NTSfsUJuDxn0Tq-svEIz

West SA, Griffin AS, Gardner A. 2007. Social semantics: altruism, cooperation, mutualism, strong reciprocity and group selection. Journal of Evolutionary Biology, 20(2): 415–432. https://onlinelibrary.wiley.com/doi/10.1111/j.1420-9101.2006.01258.x

Wilson EO. 1975. Sociobiology: The New Synthesis. Harvard University Press, USA. https://www.hup.harvard.edu/books/9780674002357

Zhang WJ. 2012. Computational Ecology: Graphs, Networks and Agent-based Modeling. World Scientific, Singapore. http://www.worldscientific.com/worldscibooks/10.1142/8118

Zhang WJ. 2015. A generalized network evolution model and self-organization theory on community assembly. Selforganizology, 2(3): 55–64. http://www.iaees.org/publications/journals/selforganizology/articles/2015-2(3)/A-network-evolution-model-and-self-organization-theory.pdf

Zhang WJ. 2016a. A random network based, node attraction facilitated network evolution method. Selforganizology, 3(1): 1–9. http://www.iaees.org/publications/journals/selforganizology/articles/2016-3(1)/a-node-attraction-facilitated-network-evolution-method.pdf

Zhang WJ. 2016b. Selforganizology: The Science of Self-Organization. World Scientific, Singapore. https://www.worldscientific.com/worldscibooks/10.1142/9685