Vicsek Swarm Model: Collective Behavior and Phase Transitions

Andrea Simone Costa

October 11, 2025

1 Introduction

The Vicsek model, introduced by Tamás Vicsek and collaborators in 1995, is a fundamental model of collective behavior in self-propelled particle systems. Unlike the previous simulations which focused on diffusion and random walks, this project explores how simple local interaction rules can lead to global coherent motion - a phenomenon observed in nature across many scales, from bacterial colonies to bird flocks and fish schools.

The model demonstrates a remarkable phase transition: at low noise levels, particles spontaneously organize into coherent groups moving in the same direction, while at high noise levels, the system remains disordered with no preferred direction. This simulation implements the 2D Vicsek model with additional behavioral rules including separation forces and boundary avoidance, providing an interactive platform to explore the emergence of collective behavior.

2 The Vicsek Model

2.1 Basic Dynamics

The Vicsek model consists of N self-propelled particles (agents) moving in a 2D domain. Each particle i has:

- Position: $\mathbf{r}_i = (x_i, y_i)$
- Direction: $\theta_i \in [0, 2\pi)$
- Constant speed: v_0 (same for all particles)

At each time step, particles update their direction according to the rule:

$$\theta_i(t+1) = \langle \theta_j(t) \rangle_{|\mathbf{r}_j - \mathbf{r}_i| < r} + \xi_i(t) \tag{1}$$

where:

- $\langle \theta_j \rangle$ is the average direction of all neighbors (including particle i itself) within radius r
- $\xi_i(t)$ is random noise uniformly distributed: $\xi_i \sim \mathcal{U}[-\eta/2, \eta/2]$
- η is the noise parameter controlling disorder

The position update is simply:

$$\mathbf{r}_i(t+1) = \mathbf{r}_i(t) + v_0 \begin{pmatrix} \cos\theta_i(t+1) \\ \sin\theta_i(t+1) \end{pmatrix} \tag{2}$$

2.2 Order Parameter

The key observable is the order parameter Φ, which measures the degree of collective alignment:

$$\Phi(t) = \frac{1}{Nv_0}\left|\sum_{i=1}^{N}\mathbf{v}_i(t)\right| = \frac{1}{N}\left|\sum_{i=1}^{N}\begin{pmatrix}\cos\theta_i\\ \sin\theta_i\end{pmatrix}\right| \tag{3}$$

The order parameter has the following interpretation:

- $\Phi = 0$: Complete disorder, particles moving in random directions
- $0 < \Phi < 1$: Partial order, some alignment but with fluctuations
- $\Phi = 1$: Perfect order, all particles moving in the same direction

2.3 Phase Transition

The Vicsek model exhibits a continuous phase transition as the noise parameter η is varied. At fixed density $\rho = N/L^2$ (where L is the system size):

- **Low noise** ($\eta < \eta_c$): Ordered phase with $\Phi > 0$, particles spontaneously align
- **High noise** ($\eta > \eta_c$): Disordered phase with $\Phi \approx 0$, random motion dominates
- **Critical noise** ($\eta \approx \eta_c$): System fluctuates between ordered and disordered states

The critical noise η_c depends on the interaction radius r and particle density ρ. The phase transition is analogous to the Ising model in statistical mechanics, though the Vicsek model describes active (non-equilibrium) matter.

3 Enhanced Implementation

This simulation extends the basic Vicsek model with two additional behavioral rules that improve realism and stability:

3.1 Separation Forces

To prevent unrealistic clustering and particle overlap, a separation force is implemented. When two particles i and j are closer than a threshold distance d_{sep}, particle i experiences a repulsive force:

$$\mathbf{F}_{\text{sep},i} = \sum_{j\neq i, d_{ij}<d_{\text{sep}}} s_{\text{sep}}\frac{d_{\text{sep}} - d_{ij}}{d_{\text{sep}}}\frac{\mathbf{r}_i - \mathbf{r}_j}{d_{ij}} \tag{4}$$

where $d_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ is the distance between particles and s_{sep} is the separation strength parameter.

This force is strongest when particles are very close ($d_{ij} \to 0$) and vanishes when $d_{ij} \geq d_{\text{sep}}$. The force is converted to an angular adjustment by calculating the desired direction of the force vector and smoothly steering toward it.

3.2 Boundary Avoidance

In systems with hard boundaries (as opposed to periodic boundary conditions), particles need to avoid colliding with walls. The simulation implements boundary avoidance by applying repulsive forces when particles approach within distance d_{avoid} of any wall:

$$\mathbf{F}_{\text{wall},i} = \sum_{\text{walls}} s_{\text{wall}} \frac{d_{\text{avoid}} - d_{\text{wall}}}{d_{\text{avoid}}} \hat{\mathbf{n}}_{\text{wall}} \tag{5}$$

where d_{wall} is the distance to the wall, $\hat{\mathbf{n}}_{\text{wall}}$ is the outward normal from the wall, and s_{wall} is the avoidance strength.

This creates smooth turning behavior near boundaries rather than abrupt reflections, resulting in more natural-looking flocking patterns.

3.3 Direction Averaging

A subtle but important detail is how directions are averaged. Naively averaging angles can lead to incorrect results due to the periodic nature of angles. For example, the average of 350 and 10 should be 0 (or 360), not 180.

The correct approach is to convert each angle to a unit vector, average the vectors, and convert back:

$$\langle \theta \rangle = \text{atan2}\left(\frac{1}{N_{\text{neighbors}}} \sum_j \sin\theta_j, \frac{1}{N_{\text{neighbors}}} \sum_j \cos\theta_j \right) \tag{6}$$

This ensures that the averaged direction correctly accounts for angle wrapping.

4 Implementation Details

4.1 Neighbor Detection

The simulation uses a brute-force $O(N^2)$ approach for neighbor detection, checking all pairs of particles to determine which are within the interaction radius. While spatial hashing or k-d trees could improve this to $O(N \log N)$ or $O(N)$, the quadratic approach is acceptable for moderate particle counts ($N \lesssim 1000$) and simplifies the implementation.

4.2 Synchronous Update

An important implementation detail is that all particle directions are calculated based on the state at time t, and then all are updated simultaneously to time $t+1$. This synchronous update ensures that no particle gets an unfair advantage by seeing updated neighbors before deciding its own direction.

The implementation achieves this by:

1. Collecting all current particle directions
2. Calculating new directions for all particles based on current state
3. Applying all new directions simultaneously
4. Updating positions based on new directions

4.3 Visualization

Particles are rendered as colored triangles pointing in their direction of motion. The color is mapped from direction using the HSL color space:

$$\text{Hue} = (\theta \times 180/\pi + 180) \mod 360 \tag{7}$$

This creates a visually appealing rainbow effect where particles moving in similar directions have similar colors, making it easy to visually identify aligned groups.

5 Observations and Analysis

5.1 Order-Disorder Transition

By varying the noise parameter η from 0 to 1, one can observe the phase transition:

- $\eta = 0$: Perfect order, $\Phi \approx 1$. All particles quickly align and move as a coherent flock.
- $\eta = 0.1$ **(default)**: High order, $\Phi \approx 0.7 - 0.9$. Particles form cohesive groups with minor fluctuations.
- $\eta = 0.5$: Transitional regime, $\Phi \approx 0.3 - 0.5$. Multiple groups form and dissipate dynamically.
- $\eta = 1.0$: Disorder, $\Phi \approx 0 - 0.2$. Particles move nearly independently with little correlation.

5.2 Effect of Interaction Radius

The interaction radius r controls the range of influence:

- **Small r (e.g., 0.5)**: Particles only align with very close neighbors. Leads to fragmented groups and lower global order. Critical noise η_c is higher (system can tolerate more noise before losing order).
- **Large r (e.g., 1.2)**: Particles interact with many neighbors. Creates more rigid, coordinated motion but can lead to globally synchronized oscillations. Lower η_c (system more sensitive to noise).

5.3 Collective Patterns

Depending on parameters, various collective patterns emerge:

- **Rotating mills**: Under certain conditions, particles can form circular rotating patterns. This occurs when particles near boundaries turn inward due to boundary avoidance, creating a self-reinforcing circular flow.
- **Streaming patterns**: In elongated domains or with anisotropic boundary geometry, particles can form coherent streams flowing in one direction.
- **Fragmentation and merging**: At intermediate noise levels, groups form, fragment, and merge dynamically - resembling the fission-fusion dynamics seen in animal groups.

6 Software Architecture

6.1 System Overview

The Vicsek swarm simulation follows a clean agent-based architecture:

```
index.ts (Entry Point)
    |
    v
VicsekModel (AgentScript Model)
    |-- turtles.create(300) -> Agent initialization
    |-- step() -> Update directions → Move agents → Calculate
    |
    +-- updateDirections() -> Alignment + forces
    +-- updatePositions() -> Ballistic motion
    |
    v
SwarmVisualization (Canvas 2D)
    |-- drawAgents() -> Triangle rendering with color mapping
```

6.2 Core Modules

VicsekModel (`vicsekModel.ts`): Extends AgentScript's `Model` class. Each agent stores position (x, y) and direction θ. `findNeighbors()` uses $O(N^2)$ brute force (acceptable for $N \leq 1000$). `calculateMeanDirection()` uses vector averaging to avoid angle wrapping issues.

SwarmVisualization (`swarmVisualization.ts`): Canvas renderer with centered coordinate system. Agents drawn as triangles using `ctx.rotate(agent.direction)` for orientation. Color mapped from direction via HSL: `hue = (direction × 180/ + 180) mod 360`.

6.3 Execution Flow

Each animation frame (`requestAnimationFrame`):

1. **Direction Update** (`updateDirections()`):
   - Find neighbors within `interactionRadius` for each agent
   - Calculate mean direction: $\bar{\theta} = \text{atan2}(\sum \sin\theta_i, \sum \cos\theta_i)$
   - Add boundary avoidance: repulsive force from walls within `boundaryAvoidanceDistance`
   - Add separation: repel from agents within `separationDistance`
   - Add noise: $\xi \sim \mathcal{U}[-\eta/2, \eta/2]$
   - Store new directions, apply simultaneously (synchronous update)
2. **Position Update** (`updatePositions()`):
   - Move forward: $x \to x + v_0 \cos\theta$, $y \to y + v_0 \sin\theta$
   - Clamp to boundaries: $x \in [-L/2, L/2]$, $y \in [-L/2, L/2]$
3. **Order Parameter**: $\Phi = (1/N)|\sum_i (\cos\theta_i, \sin\theta_i)|$
4. **Rendering**:
   - Clear canvas
   - Apply coordinate transform (translate to center)
   - For each agent: draw triangle pointing in direction with HSL color

6.4 AgentScript Integration

Key AgentScript patterns:

```
// Model setup
super({ minX: -12, maxX: 12, minY: -12, maxY: 12 })

// Create agents
turtles.create(numAgents, (agent) => {
  agent.x = (Math.random() - 0.5) * worldSize
  agent.direction = Math.random() * 2 * Math.PI
})

// Iterate over agents
turtles.ask((agent) => { /* update logic */ })
```

6.5 Canvas Rendering

Triangle rendering with rotation:

```
ctx.save()
ctx.translate(agent.x, agent.y)
ctx.rotate(agent.direction)
ctx.beginPath()
ctx.moveTo(size, 0)              // tip
ctx.lineTo(-size/2, size/2)      // bottom
ctx.lineTo(-size/2, -size/2)     // top
ctx.closePath()
const hue = ((agent.direction * 180/PI + 180) % 360
ctx.fillStyle = `hsl(${hue}, 80%, 60%)`
ctx.fill()
ctx.restore()
```

Color creates rainbow effect: agents moving in similar directions have similar colors.

6.6 Vector Averaging for Directions

Correct approach to avoid angle wrapping:

```
sumSin = sum(sin(theta_i))
sumCos = sum(cos(theta_i))
meanDirection = atan2(sumSin/N, sumCos/N)
```

This ensures 350 and 10 correctly average to 0 (not 180).

7 Connection to Biology and Active Matter

The Vicsek model, despite its simplicity, captures essential features of collective motion in biological systems:

- **Fish schools and bird flocks**: The alignment rule mimics the tendency of animals to match the direction of nearby neighbors. The separation force prevents collisions.

- **Bacterial colonies**: Some bacteria exhibit collective swimming patterns driven by hydrodynamic and chemical interactions, qualitatively similar to Vicsek dynamics.
- **Crowd dynamics**: Human pedestrian motion can exhibit emergent patterns resembling Vicsek-type alignment, though with more complex decision-making.

The key insight is that global order emerges from purely local interactions - no leader or central coordination is needed. This is a fundamental principle of self-organization in complex systems.

8 Conclusion

The Vicsek model demonstrates how simple interaction rules can generate complex collective behavior. The simulation illustrates the phase transition between ordered and disordered states, the role of noise in disrupting coordination, and the emergence of patterns from local interactions. By extending the basic model with separation forces and boundary avoidance, the simulation produces more realistic flocking behavior suitable for understanding biological and artificial swarm systems. This project serves as an accessible introduction to active matter physics and self-organization, concepts at the frontier of modern statistical physics and complexity science.