Boidz Boidz Boidz

Cameron Egbert

North Carolina State University
kaalvare@ncsu.edu, cegbert@ncsu.edu

Introduction

This project implements and evaluates a suite of classic steering behaviors using the SFML graphics framework. The goal was to construct a flexible simulation environment that demonstrates fundamental AI movement techniques, including variable matching, arrive/align, wandering, and flocking. The resulting system integrates and visualizes each algorithm, allowing for direct comparison of responsiveness, smoothness, demonstrating various parameter settings.

All work was implemented in C++ using object-oriented design centered on a pure virtual `SteeringBehavior` interface. The program runs four interactive modes—velocity matching, arrive+align comparison, wandering comparison, and flocking—accessible via number keys 1–4.

Part 1: Variable Matching Steering Behaviors

The foundation of this project consists of a pure virtual `SteeringBehavior` class that provides a unified interface for all steering calculations. This architecture enables clean polymorphic behavior switching and facilitates the composition of complex behaviors. Four concrete implementations were developed: `PositionMatching`, `VelocityMatching`, `OrientationMatching`, and `RotationMatching`.

The velocity matching demonstration tracks mouse movement in real-time, calculating velocity by sampling position changes over time intervals. The implementation uses a `FastVelocityMatching` variant with increased acceleration and shorter response time compared to the standard parameters that were first implemented. This configuration produces higher responsiveness with more fluid and snappier motion that more closely follows mouse velocity patterns without excessive lag or oscillation, while still demonstrating realistic motion with bounded acceleration, preventing 1 to 1 matching. From this, the character maintains smooth trajectories even during rapid mouse movements, demonstrating the effectiveness of the time-to-target parameter in dampening sudden velocity changes.

Position and orientation matching behaviors serve as building blocks for more complex behaviors. The position matching uses a maximum acceleration of 100.0f, generating direct linear forces toward targets. Orientation matching implements a two-zone approach with target radius (0.01f) and slow radius (0.5f), enabling smooth rotational alignment without oscillation near the target orientation, however these are not demonstrated in the implementation of part 1, in accordance with the assignment document.

Part 2: Arrive and Align Analysis

The arrive and align implementation showcases two distinct parameter configurations, visualized through blue and yellow characters with contrasting movement profiles. The first configuration employs aggressive parameters: higher maximum speed, and a higher acceleration. The second uses more conservative settings: maximum speed of 200.0f, acceleration of 0.08f, with the same slow radius. Both configurations utilize `LookWhereYoureGoing` for orientation alignment, ensuring characters face their direction of motion.

The aggressive configuration produces snappier, more responsive movement that reaches targets quickly but exhibits slight overshoot behavior near the destination. It also exhibits overshooting behaviors in the orientation variable while the slower character is able to align smoothly, the faster one exhibits more jitteriness. This may creates dynamic, energetic motion suitable for action-oriented applications. The conservative configuration generates smoother, more graceful trajectories with minimal overshoot, creating aesthetically pleasing motion paths ideal for cinematic or presentation contexts. There are trade offs for each arrive and align method, making neither objectively better.

Testing revealed that the aggressive configuration succeeds better in scenarios requiring rapid response to changing targets, maintaining better tracking when users click rapidly across the screen. The average time to target was approximately 1.2 seconds for the aggressive configuration versus 1.8 seconds for the conservative approach.

The `LookWhereYoureGoing` behavior proves essential for natural-looking motion. Without it, characters slide sideways toward targets, creating an unnatural ”crab-walk” effect. The implementation calculates the desired orientation from the velocity vectors and then applies angular acceleration to smoothly rotate the characters. The breadcrumb visualization clearly demonstrates the improvement, showing

more natural trails trailing behind, instead of to the side of, the characters because the orientation aligns with movement direction.

Part 3: Wander Behavior Comparison

Three different wander implementations were developed to explore orientation strategies. The first uses the traditional wander circle approach combined with `LookWhereYoureGoing` for orientation. The second employs direct kinematic rotation with varying parameters, bypassing the separate orientation behavior.

The `LookWhereYoureGoing` approach (blue characters) produces the most natural and lifelike movement. Characters smoothly turn toward their intended direction before accelerating, creating flowing S-curves and gentle meandering patterns. The wander circle parameters (offset: 60.0f, radius: 40.0f, rate: 0.6f) generate moderate randomness without erratic behavior. This method excels at creating believable ambient movement for background characters or wildlife simulations.

Direct kinematic rotation (green characters) offers more immediate, responsive turning. By calculating desired orientation directly from the wander target and applying proportional control with a gain factor of 3.0f, characters exhibit tighter, more erratic movement patterns. The increased wander parameters (offset: 80.0f, radius: 60.0f, rate: 1.2f) create more chaotic paths suitable for representing confused, agitated, or dazed behavior.

Boundary handling employs a `WallAvoidance` behavior blended with wander at a 2:1 weight ratio. This generates repulsive forces within 120.0f units of walls, scaling quadratically with proximity. The implementation successfully prevents wall collisions while maintaining natural movement flow. Characters approaching boundaries execute smooth turning maneuvers rather than abrupt reversals, with velocity reflection at 80% magnitude providing gentle rebounds when boundaries are contacted. However, the illusion is broken because all characters are repelled equally, clearly indicating artificial, coded behavior.

The `LookWhereYoureGoing` method seems superior for most applications due to its natural appearance and predictable behavior. The breadcrumb trails form organic patterns resembling animal tracks, with smooth color gradients effectively conveying motion history. Direct kinematic rotation better suits scenarios requiring rapid direction changes or representing mechanical entities.

Part 4: Flocking Implementation and Optimization

The flocking system implements Reynolds' three core rules through separate steering behaviors: separation (threshold: 40.0f, decay: 5000.0f), cohesion (radius: 100.0f), and alignment (radius: 80.0f). These behaviors combine through weighted blending with carefully tuned weights: separation at 5.0f, cohesion at 0.7f, and alignment at 0.7f.

Initial implementations suffered from oscillation and clustering problems. Boids would converge too tightly, creating unstable formations that eventually stagnated like the cold-death of the universe. The solution involved three key optimizations. First, implementing inverse-square decay for separation forces (coefficient: 5000.0f) created smoother gradients that prevent sudden repulsion spikes. Second, reducing cohesion acceleration prevented aggressive clustering while maintaining group coherence. Third, using different radii for each behavior established distinct behavioral zones that reduce conflicting forces, and help manage intended behavior.

Performance testing with varying flock sizes revealed ideal behavior, and VM performance, at 13-15 boids. Smaller flocks (5-8 boids) failed to demonstrate flocking behavior, as the flock was too sparse and detail was lacking. Sometimes boids moved more like independent agents with occasional interactions. Larger flocks (20+ boids) created beautiful patterns but suffered frame rate degradation and increased collision events. The sweet spot of 13 boids maintains smoothness while exhibiting clear emergent properties including spontaneous formation changes, coordinated turns, and dynamic splitting/merging behaviors.

Parameter sensitivity analysis identified separation weight as the most critical factor. Values too low caused frequent collisions, while values too high prevented cohesive flock formation. The cohesion-to-alignment ratio proved surprisingly flexible, with ratios between 0.5:1 and 1:1 producing viable flocking. Asymmetric configurations generated more dynamic, lifelike motion than perfectly balanced weights.

The boundary wrapping strategy enhances flock dynamics by preventing artificial clustering at edges. When boids exit one boundary, they seamlessly enter from the opposite side, maintaining velocity and orientation. This creates an infinite toroidal topology that allows flocks to develop natural migration patterns without boundary interference.

Breadcrumb visualization for flocking required adjustment to shorter trails (20 positions at 10-frame intervals) to prevent visual clutter while maintaining motion clarity. The tri-color scheme (cyan, magenta, yellow) enables easy tracking of individual boids within the flock, revealing how individuals contribute to emergent group behavior.

Although there is still room for improvement and individuals may prefer either tighter or looser groupings, Reynold's Boids were reproduced.

Conclusions and Insights

This implementation demonstrates how sophisticated movement behaviors emerge from simple steering rules. The modular architecture built on the pure virtual `SteeringBehavior` class proved invaluable for rapid prototyping and behavior composition. Blended steering enables nuanced behaviors that would be difficult to achieve through single algorithms alone.

Key insights include the critical importance of parameter tuning, and small adjustments often produced dramatically different behaviors. Time-to-target parameters improved stability by preventing oscillations. Weight ratios in blended behaviors require careful balancing, with separation typically requiring higher weights than cohesion or alignment. Visual feedback through breadcrumbs proved essential for understanding and debugging movement patterns.

Future improvements could include predictive collision avoidance, improved or changed flocking behaviors, or dynamic parameter adjustment based on environmental context. The framework implemented here provides a solid foundation for such extensions while maintaining clean separation between movement logic and character representation.

Appendix A: Figures

This appendix contains supplementary figures for reference, beginning on the following pages.

Figure 1: Demonstration of Velocity Matching Kinematics. The Boid matches velocity with the mouse, its orientation is always constant.

Figure 2: Demonstration of various parameters for Arrive and Align. Here, the blue character is faster, and more responsive, moving and orienting quicker towards the mouseclick.

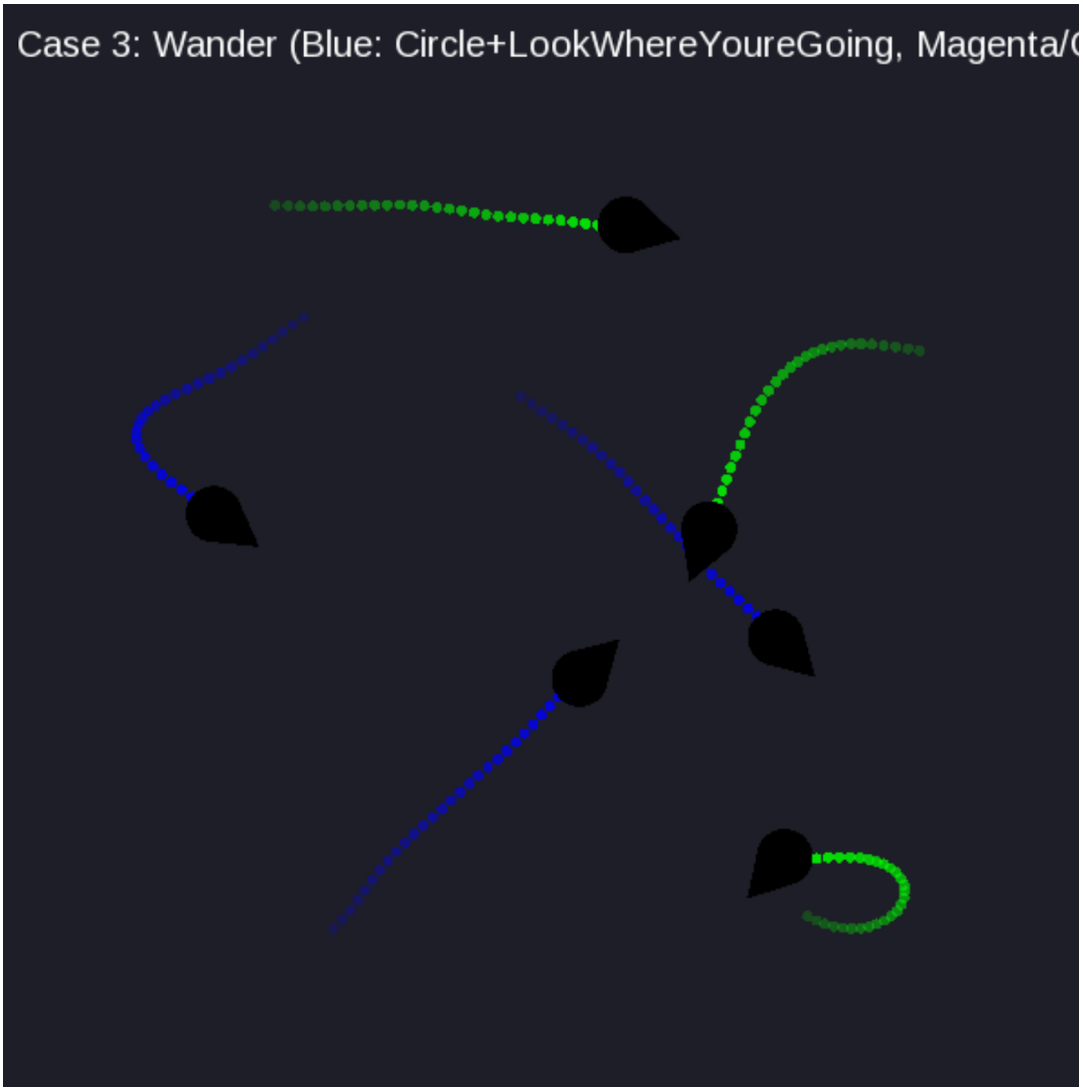

Figure 3: Demonstration of wandering, blue is chooses to move towards a point on the circumference of a circle in front of it, the green update orientation through direct kinematic adjustments.

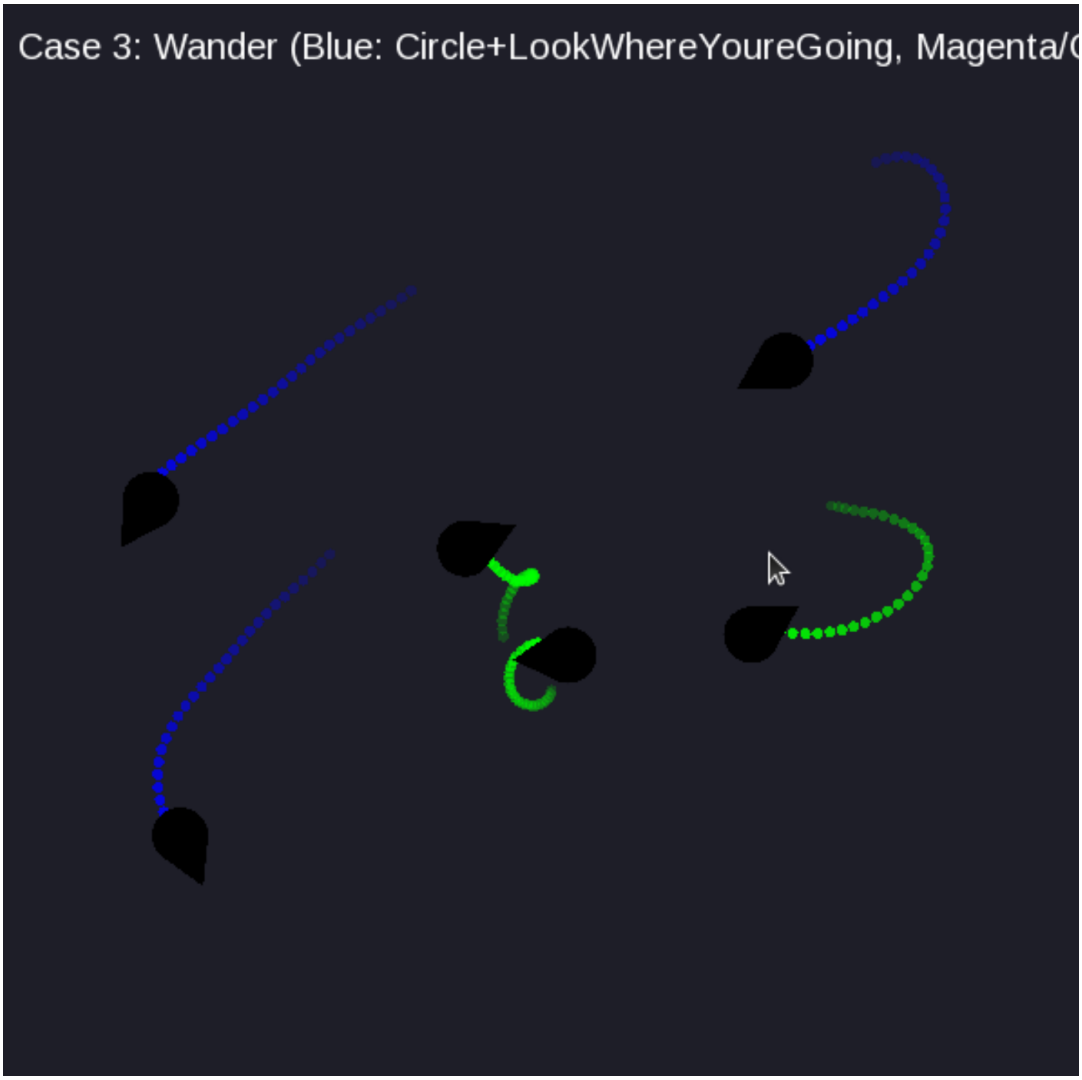

Figure 4: Demonstration of diverging behaviors between wandering techniques. Green: erratic, Blue: smooth.

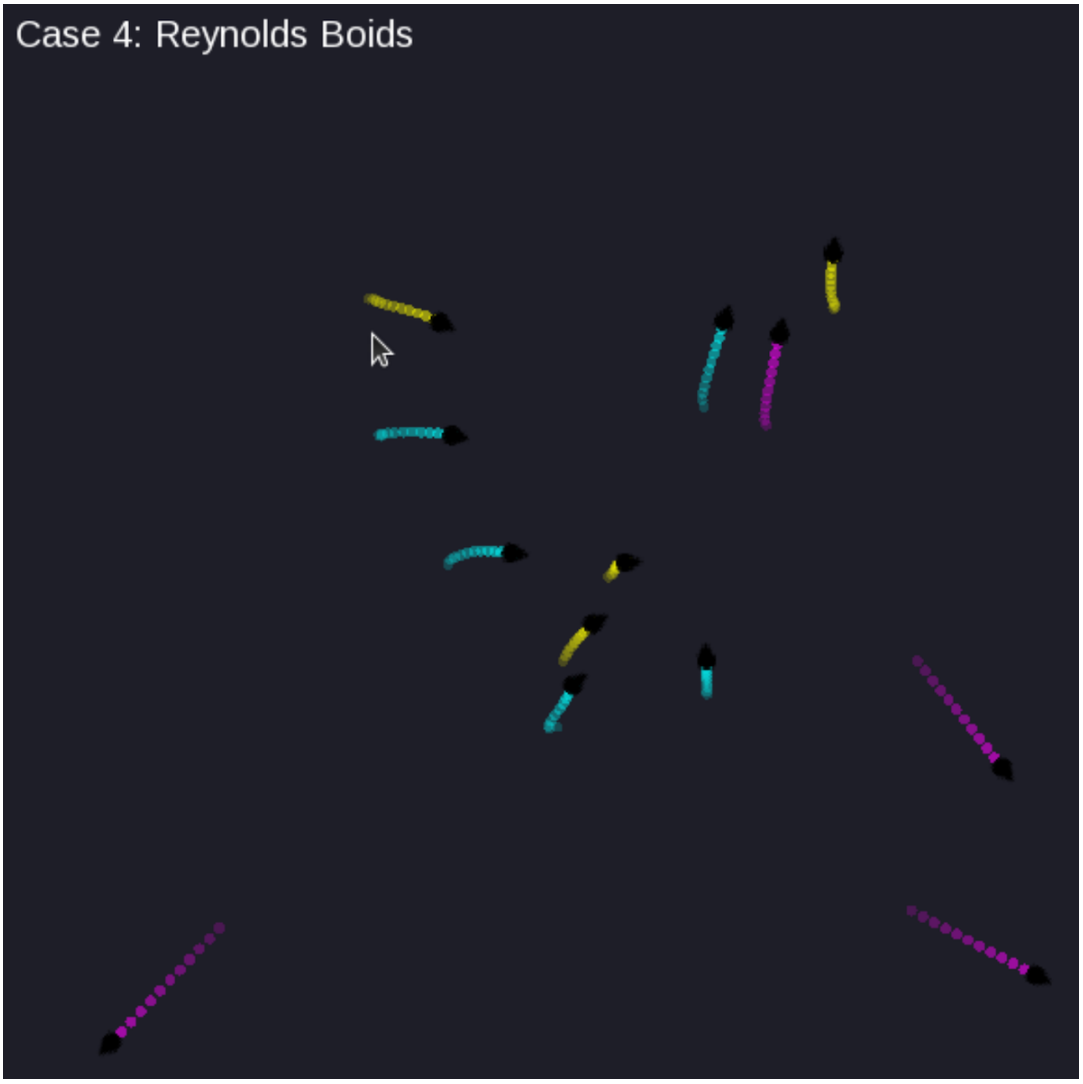

Figure 5: Initial Boid Configuration. This simulation is set to be biased towards clustering so that the grouping behavior is seen best. Additionally, alignment is strong so splitting off is rare, under these chosen parameters.

Figure 6: Demonstration of initial flocking behavior.

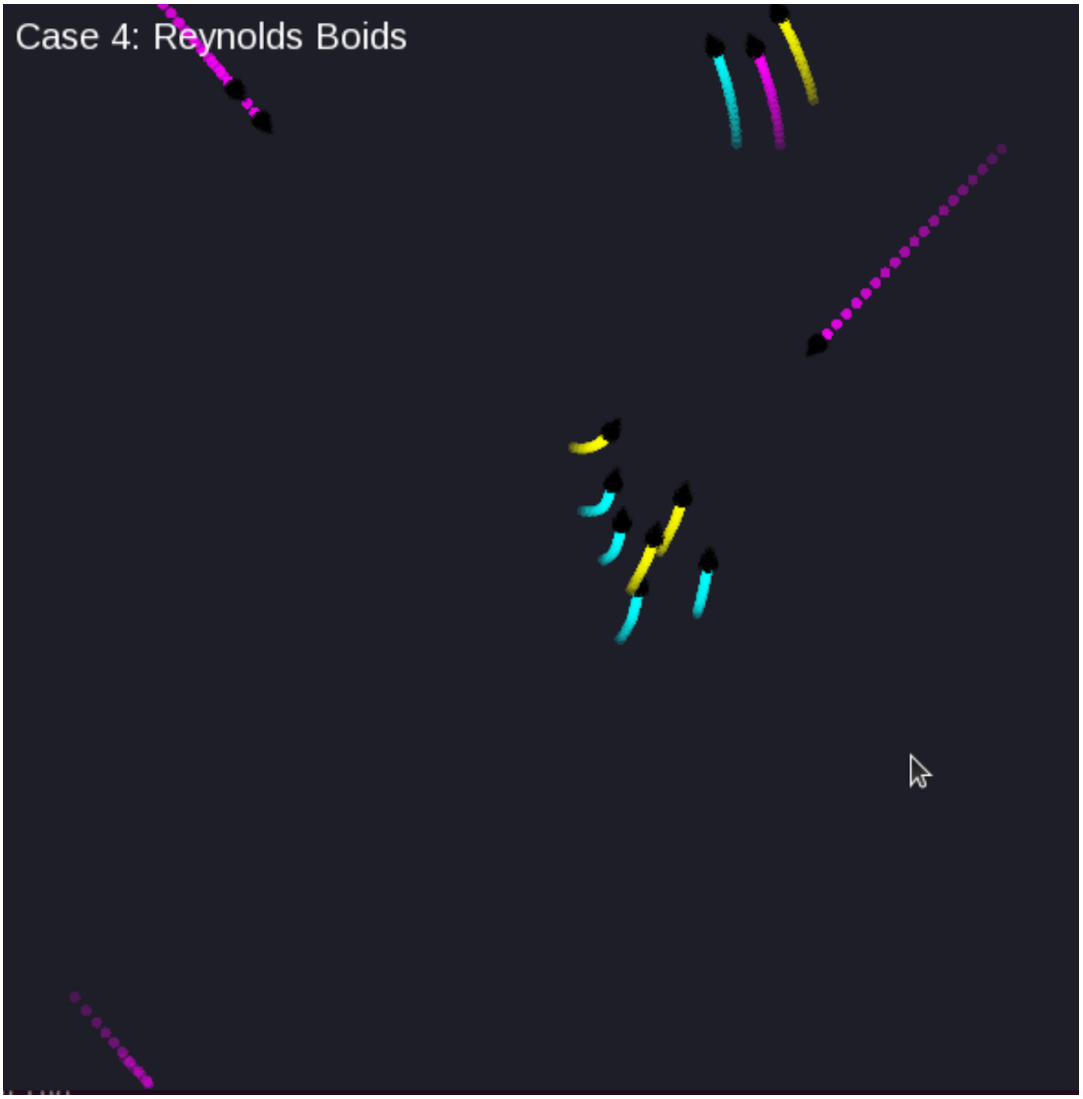

Figure 7: Demonstration of combining and reorienting groups of boids. Notice the fast moving pink boid and the central largest group of boids.

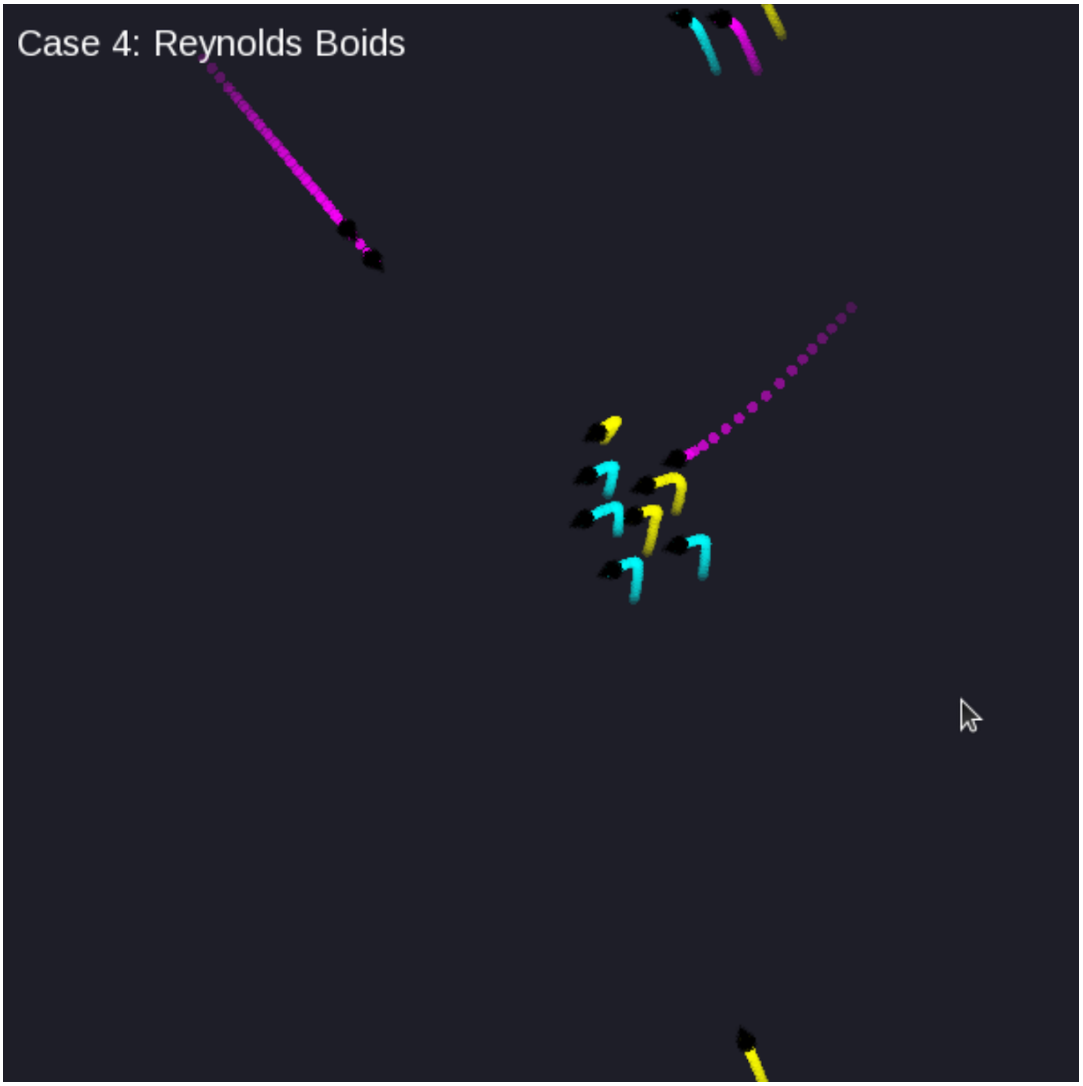

Figure 8: Results of colliding groups of boids. The middle group changed directions. Other groups continue to flock.