**Evaluation of Steering Behaviors**

**Aditi Reddy**

**Unity id: areddy5**

**Part 1:**

For the first part of the assignment, I built a library that encapsulated four variable matching steering behaviors. The approach was to define a pure virtual class, SteeringBehavior, which served as the base for all behavior types. I then created four subclasses to handle position, orientation, velocity, and rotation matching separately. The velocity matching behavior was made such that the boid matches the speed and direction of the mouse pointer. This was done by sampling the mouse pointer's location at periodic intervals, computing the velocity by dividing the change in position by the elapsed time, and then encapsulating that value in a kinematic data structure. The parameters involved here mainly included the sample interval (which indirectly controlled the precision of the velocity estimation) and the conversion factors necessary for translating screen coordinates into movement vectors.

**Part 2:**

In part 2 the arrive behavior was initialized with parameters such as maximum acceleration, maximum speed, target radius, slow radius, and a time-to-target factor. Initially, when high rotation speeds and high angular accelerations were used, the boid would display erratic behavior—it would squiggle when it tried to approach the target, and also fail to decelerate properly because of low target radius and slow radius values. Similarly, the align behavior suffered when the rotation speed was set too low, causing the boid to fail to align accurately within the available time. After some experimentation, I updated the arrive parameters to (300.f, 250.f, 5.f, 200.f, 0.5f) and the align parameters to (18.f, PI/1.f, 0.05.f, 0.5.f, 0.1.f). These adjustments resulted in a much smoother approach: the boid decelerated cleanly as it neared the target position, and it aligned its orientation accurately in the available time. Balancing these parameters was critical—the maximum acceleration and speed needed to be high enough to cover the distance quickly, while the target and slow radii, along with the time-to-target, ensured a gentle deceleration and a seamless transition into alignment.

To further refine this behavior, extensive testing revealed that even slight tweaks could significantly affect the boid's motion. The interplay between acceleration and deceleration was crucial; if the boid began to slow too early or too late, it would either overshoot or hesitate near the target. I observed that increasing the slow radius provided a longer buffer zone for deceleration, allowing the boid to modulate its speed more effectively. Additionally, adjusting

the time-to-target factor helped synchronize the boid's rotational alignment with its linear movement, ensuring that it faced the target just as it arrived. Adjusting the maximum speed was essential for maintaining momentum without sacrificing control, and each parameter influenced the others in subtle ways.

**Part 3:**

For the wander behavior, the objective was to generate a natural, unpredictable motion that still looked controlled. The initial parameters—(50.f, 100.f, 20.f, 100.f, 2.f, 0.1.f)—resulted in a boid that almost looked like it was malfunctioning: it rotated in place and moved very slowly, disrupting the illusion of organic wandering. By decreasing the wander radius and wander rate, and updating the parameters to (50.f, 100.f, 150.f, 30.f, 0.5.f, 0.2.f), the algorithm began calculating wander points more effectively. The boid's movement became smoother and more fluid, which better simulated a natural wandering pattern. I also tried different methods for handling the boid's orientation changes. One approach involved allowing the boid to continuously change its orientation based on calculated wander points, while another method employed a more constrained rotation that limited excessive turning. I found that the latter, more constrained method yielded a more aesthetically pleasing result, as it prevented sudden, jerky movements. I also revised the boundary handling strategy. My first attempt used a bounce-off mechanism at the edges of the screen, but this led to jerky, abrupt direction changes that detracted from the overall smoothness. Instead, I implemented a "snakes game" style boundary condition, where the boid reappears on the opposite side when it leaves the screen.

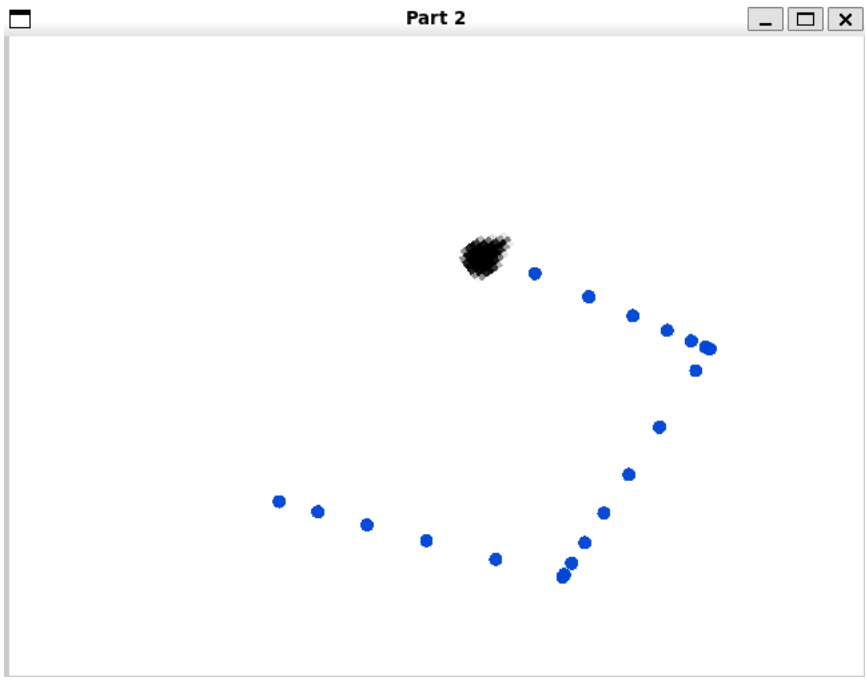
Expanding further on the wander behavior, I discovered that tweaking the wander rate had a significant impact on the natural feel of the movement. A lower wander rate allowed the boid to gradually shift its heading, resulting in a more lifelike meandering path. I experimented with various wander distances to find an optimal value where the boid neither overshot its wandering circle nor appeared too static. Incorporating a slight randomness to both the magnitude and direction of the wander vector added subtle unpredictability while maintaining control. The constrained orientation adjustment proved effective at curbing erratic spins, ensuring the boid's rotations were smooth and predictable. By fusing these adjustments with the modified boundary conditions, the overall motion felt cohesive and continuous. I also monitored the boid's response time to orientation changes, ensuring that even during rapid wander sequences, the motion remained fluid.

**Part 4:**

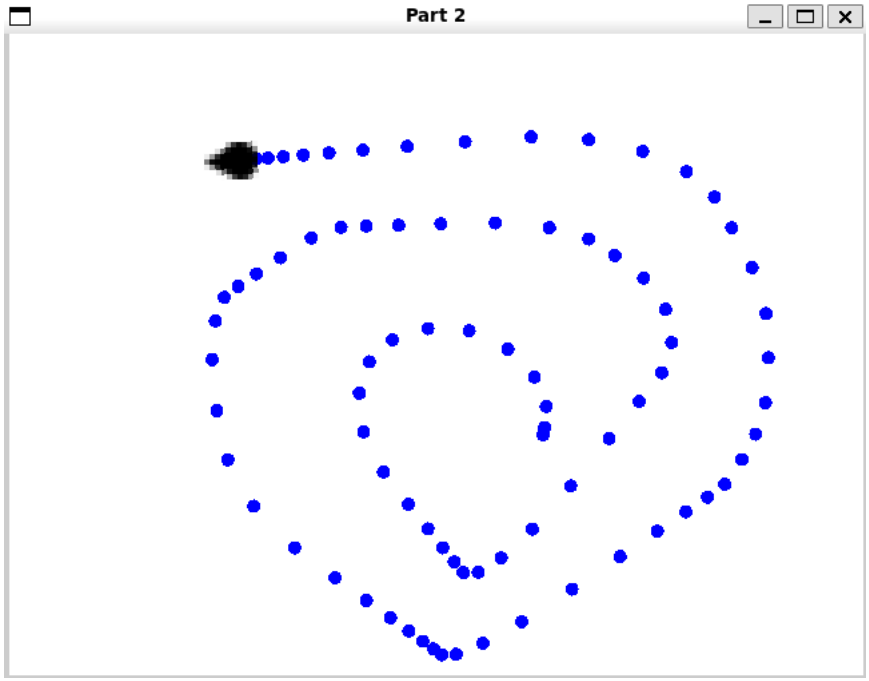
Flocking behavior integrates the individual steering behaviors into a cohesive group dynamic. Initially, I set the parameters with a neighbor radius and separation radius of 20.f, a separation weight of 5.0f, alignment weight of 1.f, cohesion weight of 1.0f, and a maximum acceleration of 250.f. While these values successfully grouped the boids together, they led to the boids all overlapping and isolated boids that would end up not getting flocked until a long long time. Many boids didn't immediately join the flock and only did so when they came very close to others.

To address this, I increased both the neighbor radius and separation radius. This change allowed each boid to detect its neighbors from a greater distance, ensuring a more consistent flocking behavior. Moreover, raising the separation weight helped to prevent overlapping by forcing boids to maintain a healthier distance from one another. The result was a more visually coherent flock, where individual units moved harmoniously and adjusted their trajectories smoothly based on their neighbors' positions and velocities.

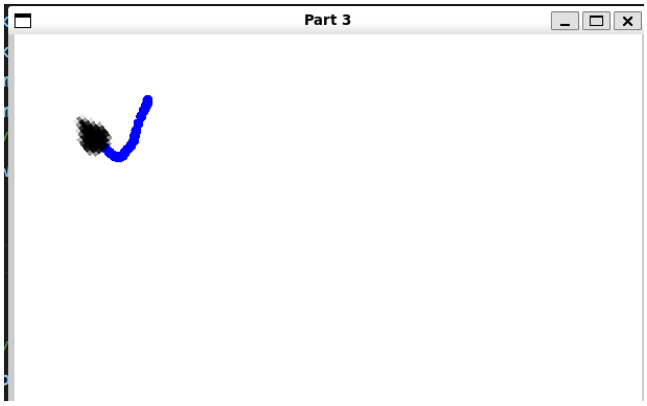
Part 2a: alignment is not proper and does not decelerate, the breadcrumbs are wide apart near target position.

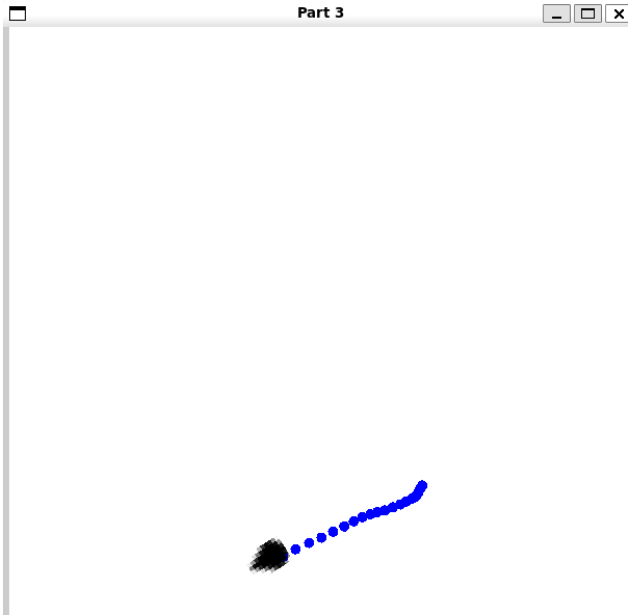


Part 2b: Boid perfectly aligns in the direction of the target location and decelerates as it reaches the target.

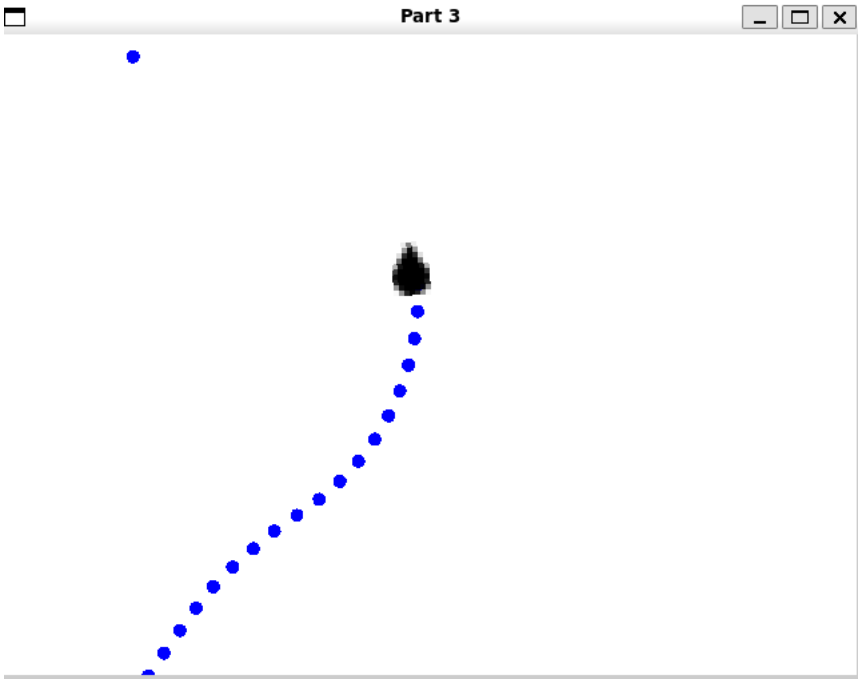



Part 3a: The boid moves very slowly and continuously takes turns in a very ant crawling type of wandering. Keeps oscillating as it moves.

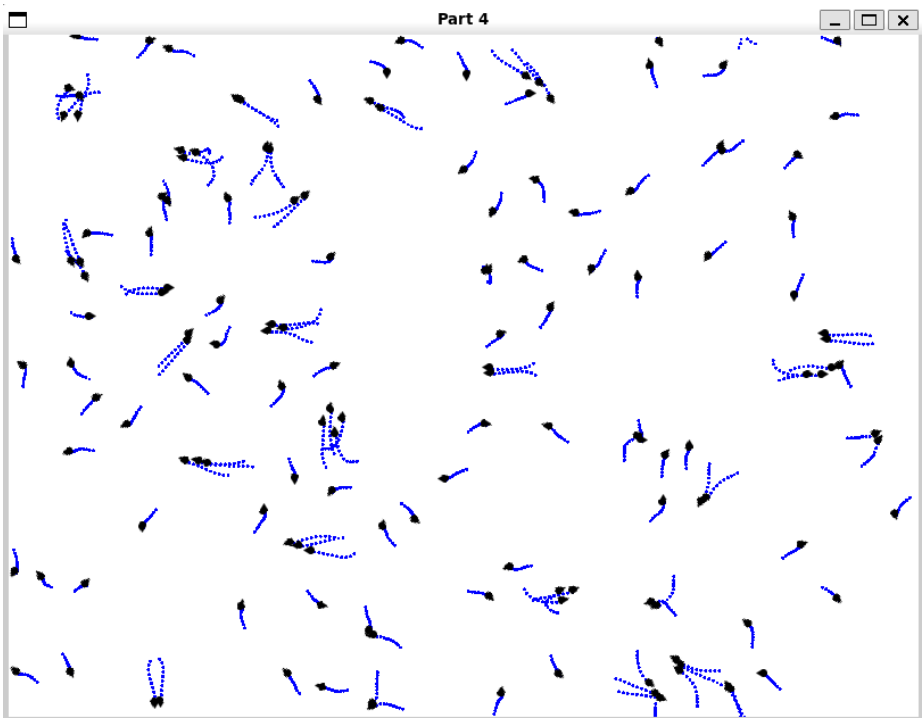


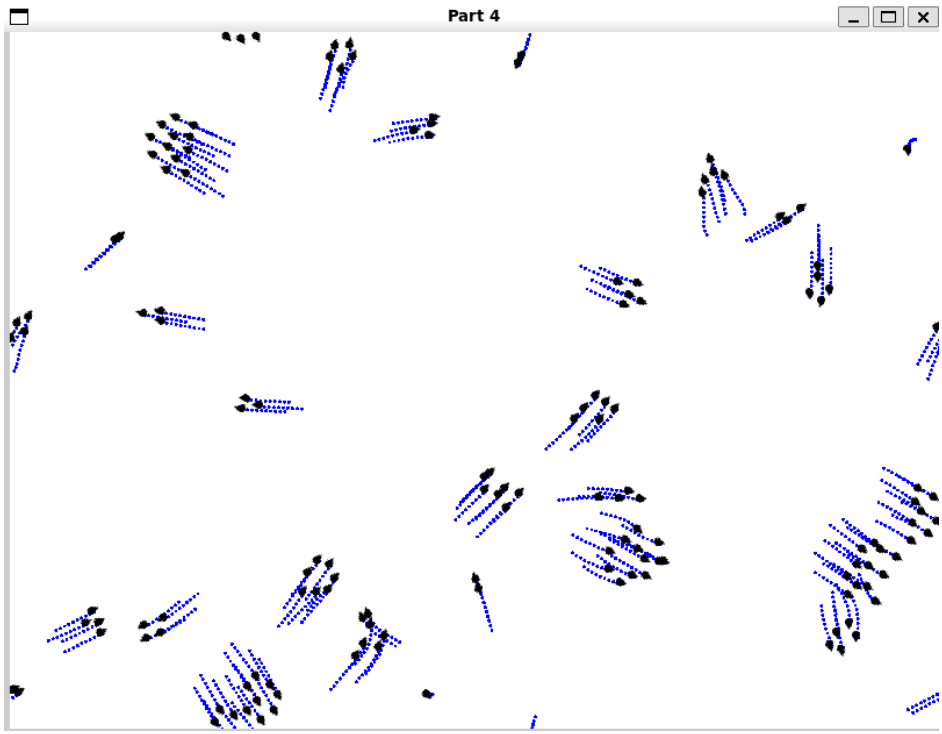



Part 3b: Wanders at a pleasing speed and does not oscillate randomly whenever it wants.

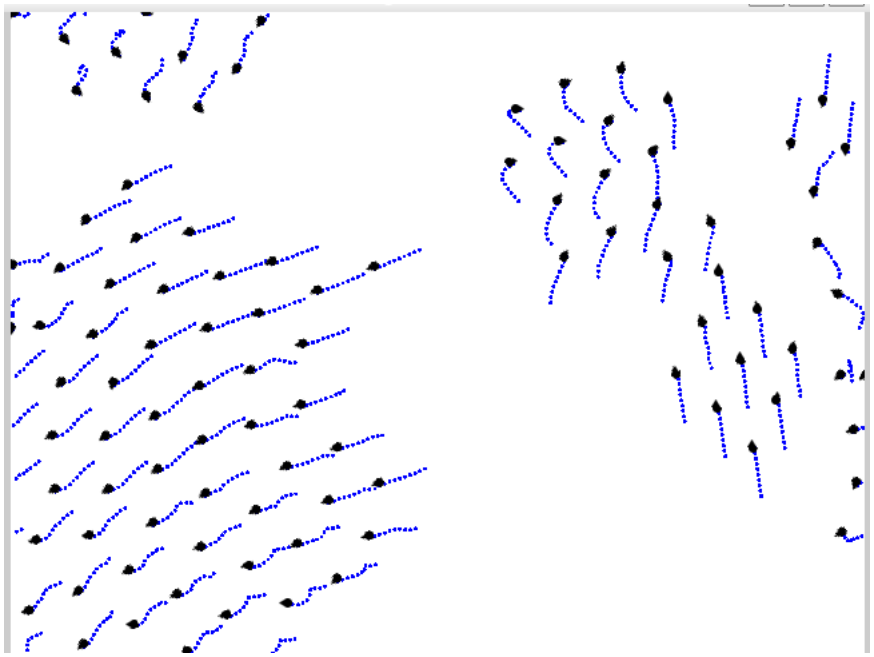



Part 4a: All the boids flock eventually, but the overlap and dash into each other and a lot of the boids are alone for a long time until they eventually flock into a big group that comes very close by.






Part 4b: All boids flock and do not overlap.



*The assistance of ChatGPT was used to make this documentation.*