CSC 484/584, Spring 2020

Assignment 1 – Movement

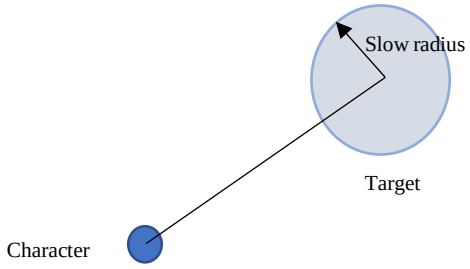
Unity id: mnagdev

1. **Kinematic Motion**

   Implemented basic kinematic motion by taking the positions of the character and the target as the input. Calculated the direction of motion by subtracting the target location from the character location and then assigned the velocity to the character in the direction of motion. I used 'PVector' data structure in processing for the position, direction, and velocity; 'float' for the angle by which the character should rotate; 'ArrayList' to store the points in the path traversed by the character. To find the orientation of the character, I used the formula atan(velocity.y / velocity.x). The character starts from the bottom left corner of the screen and moves towards the bottom right of the screen. It traverses the edge of the screen in an anti-clockwise direction and stops when it arrives at the starting location.

2. **Arrive Steering Behaviors**

   a. First method: The target location is set by mousePressed() method. My character slows down when it reaches close to the target. When the distance between the character and the target becomes less than or equal to the 'slow radius' then the speed of the character becomes proportional to the distance between the target and the character. The velocity of the character becomes zero when it is very close to the target.



   b. Second method: Here, first I calculated the desired velocity by subtracting the character's position from the target position. Then, I found the steering velocity by subtracting the character's current velocity from the desired velocity. Now, I added steering velocity to the character's velocity to turn the character in the direction of the target.

   I think that the character's movement from the first method looks better. In the second method the character oscillates around the target and this is not the case with the first method. The motion of the character is smoother in the first method.

3. **Wander Steering Behaviors**

   a. First method: For wander steering behavior, I considered that there is an imaginary circle at some fixed distance from the character and the target is somewhere on the periphery of the circle. I calculate the position of the center of the circle based on the position of the character. The orientation of the target changes once in 20 game loops

and so the wander behavior doesn't look abrupt. The location of the target is determined by its orientation with respect to the center of the circle. Then, I apply the seek algorithm to move the character towards the target. Whenever the character collides with the boundary of the screen, its velocity component is reversed.

b. Second method: In this method, the target orientation is the sum of the character's current orientation and the random generated orientation value. The target orientation is calculated in every game loop. The character moves in the direction of the target location.

Both the methods have different outputs and it depends on the situation to select which one is better. The motion of the character in the first method is more randomized than in the second method. So, if one wants the motion of the character to look like the character is confused as to where to go then the first method is more appropriate but if one wants that the motion of character must be somewhat random and not completely off the track then the second method is a better approach.

4. Flocking Behavior and Blending/Arbitration

I referred the standard Boids algorithm to demonstrate the flocking behavior. I used separation, alignment, cohesion, and seek forces.

a. Separation: It takes separation distance as the parameter value and then compares the distance between every pair of boids with this separation distance. If the distance between the boids less than the separation distance, then the velocity of the boids is directed away from each other.

b. Alignment: This means that the boids in a group have the same alignment. To perform alignment, we take the average of the velocities of the boids that are in the same group and then the steer velocity equals the difference of the boid velocity and the average velocity of the boids in the group.

c. Cohesion: This is the force that brings the neighboring boids in a group. All the boids seem to seek a common target. So, in cohesion we find the average of the positions of all the boids in a group and then make them seek this location.

I calculated the steer in each of these methods and then multiplied each force with the appropriate weights to find the resultant force acting on the boid. The boids seek the target location which is the mouse location.

When the number of followers is high the output looks chaotic. All the boids seem to accumulate at the target location and then disperse in random direction. The behavior of the flock is more clearly visible with around 100 to 200 boids.

When there are two wanderers, the boids follow the wanderer which is closer to them. I used two colors to identify the two groups of boids. The color of the boid changes based on the wanderer it is following. The wanderers are bigger in size than the boids and have a brighter color than the boids. To implement this behavior, I compared the distance of every boid with the wanderers and the wanderer which is near to the boid becomes its target. All the forces, separate, cohesion, and alignment work the same way as they work in the simple boid flocking behavior.

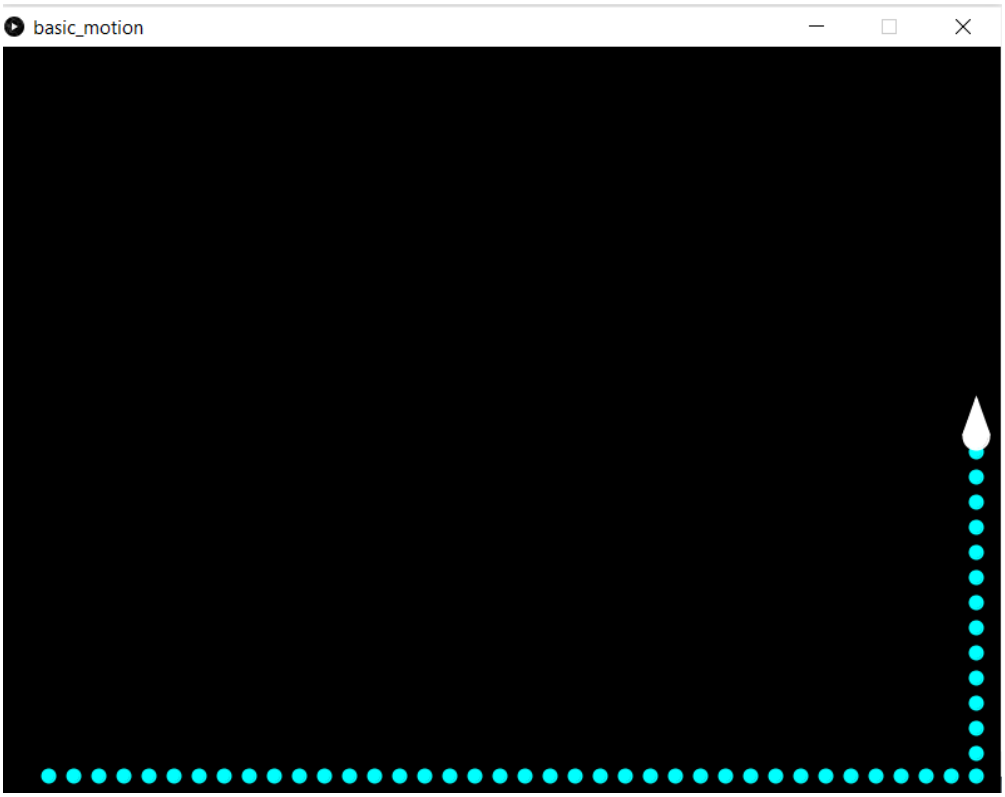


Figure 1: Basic_Motion

Figure 2: Basic_Motion

Figure 3: Basic_Motion – Character stops after reaching the starting point

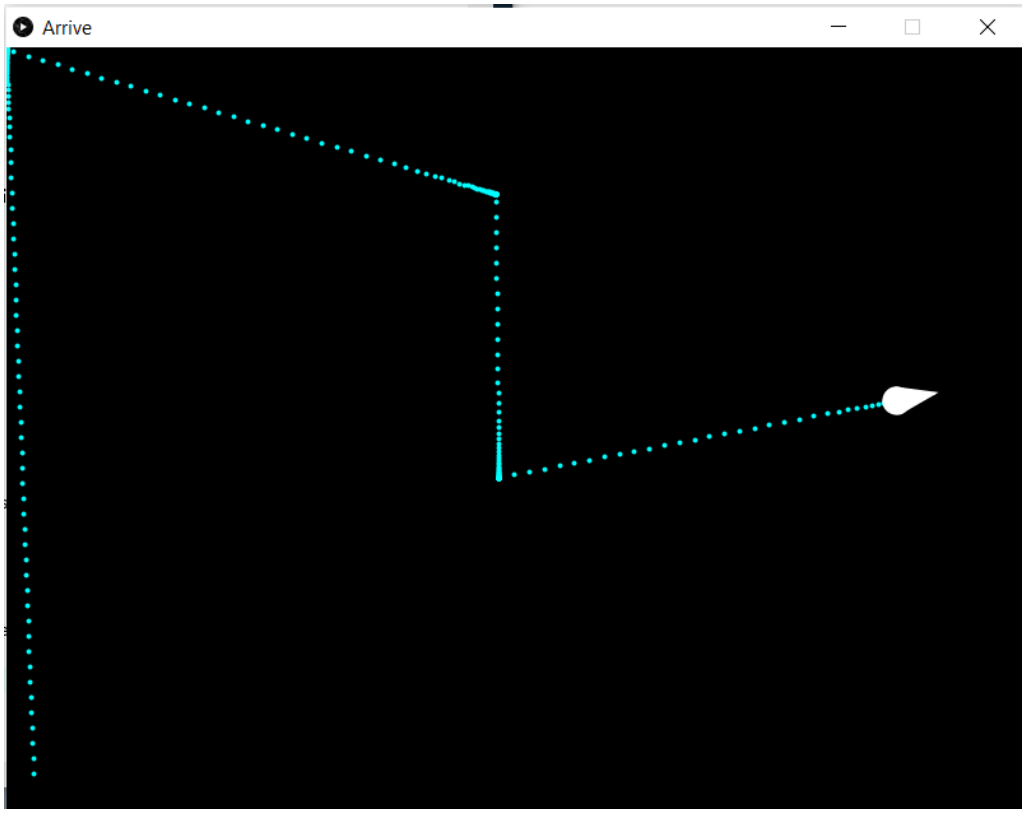


Figure 4: Arrive Steering Behaviors First Method

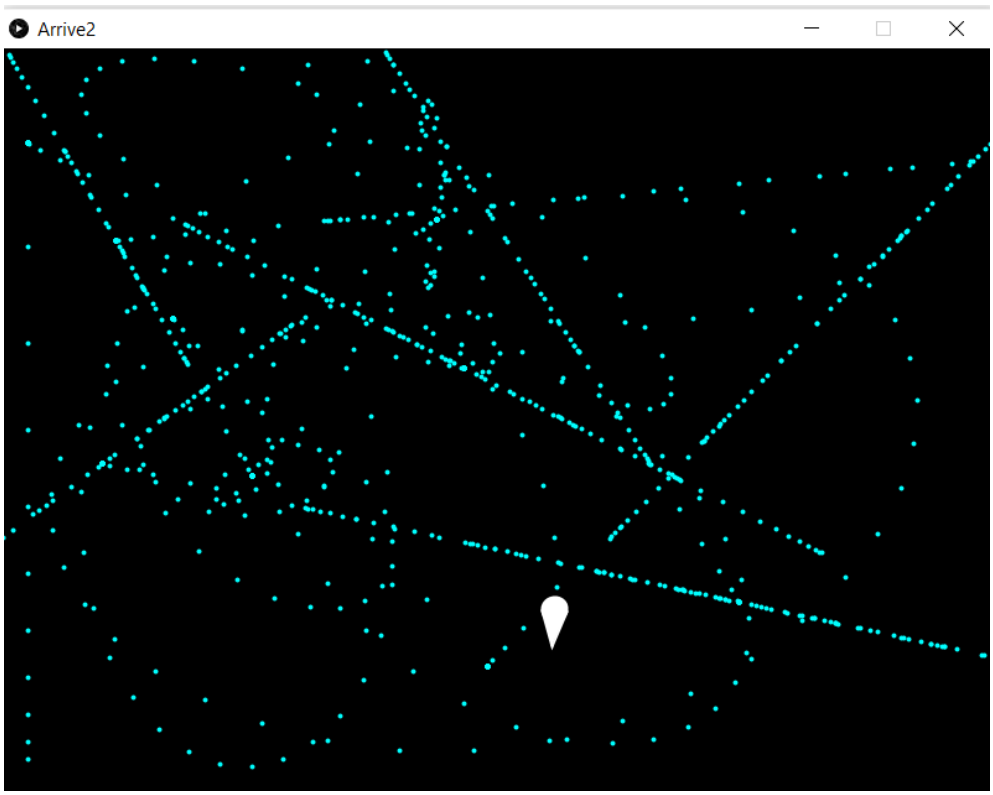


Figure 5: Arrive Steering Behaviors Second Method

Figure 6: Wander Steering Behaviors First Method

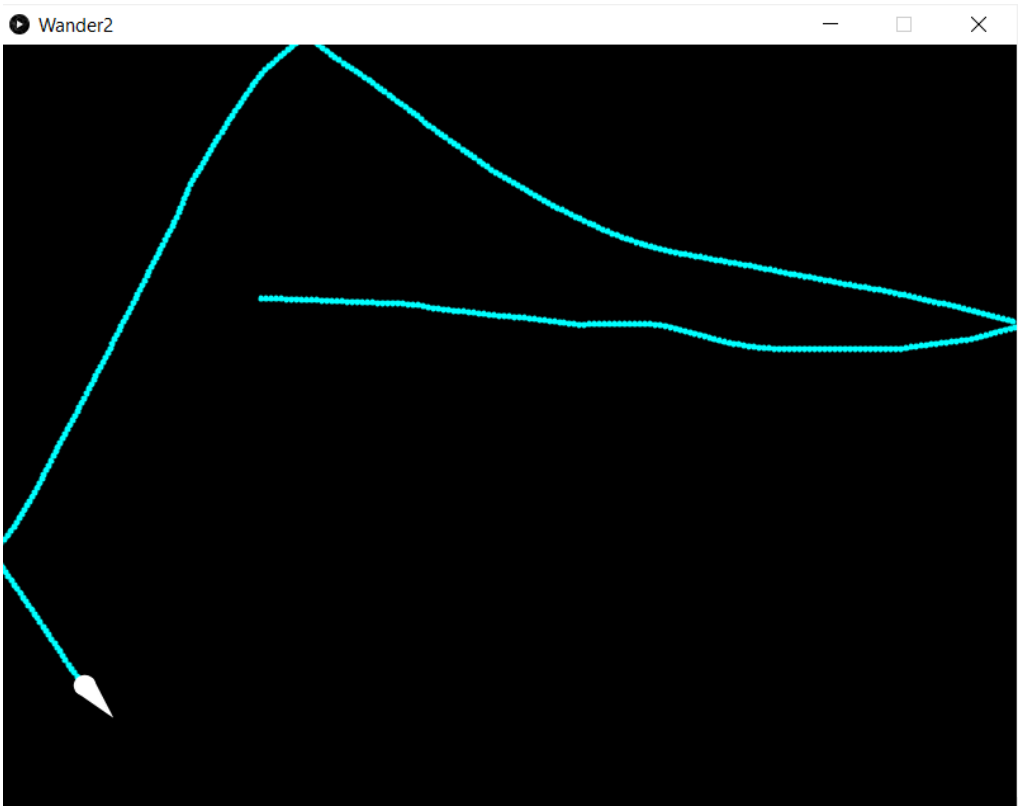


Figure 7: Wander Steering Behaviors Second Method

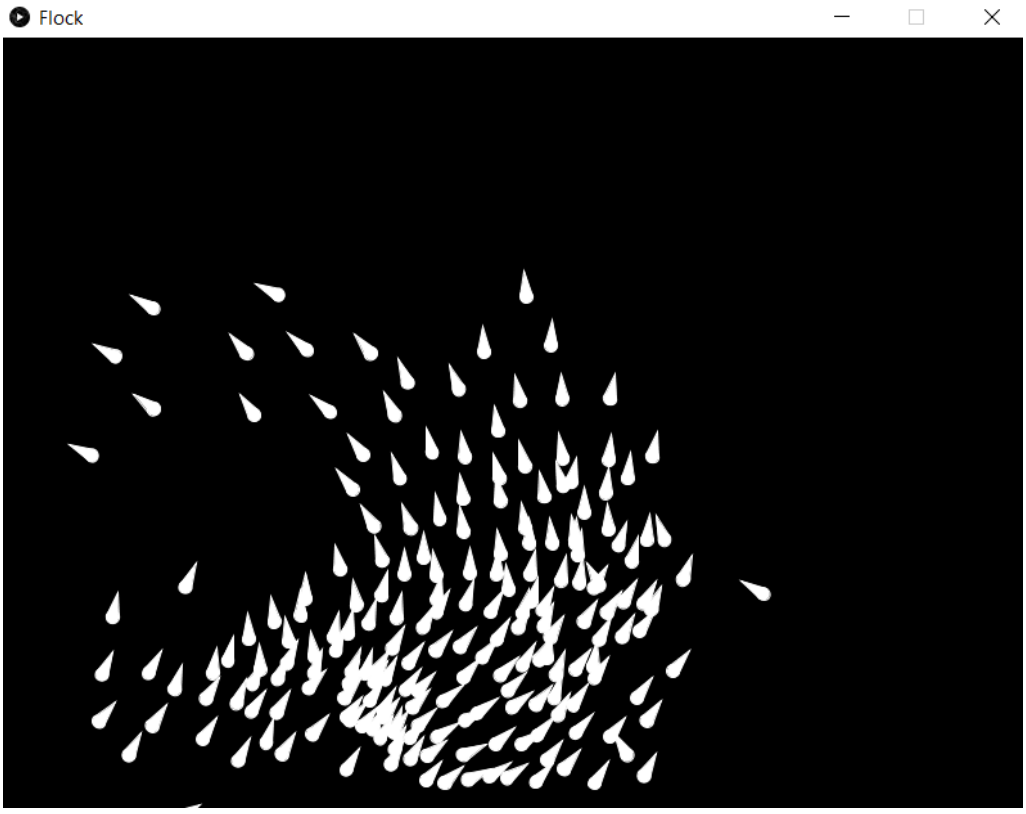


Figure 8: Flocking Behavior

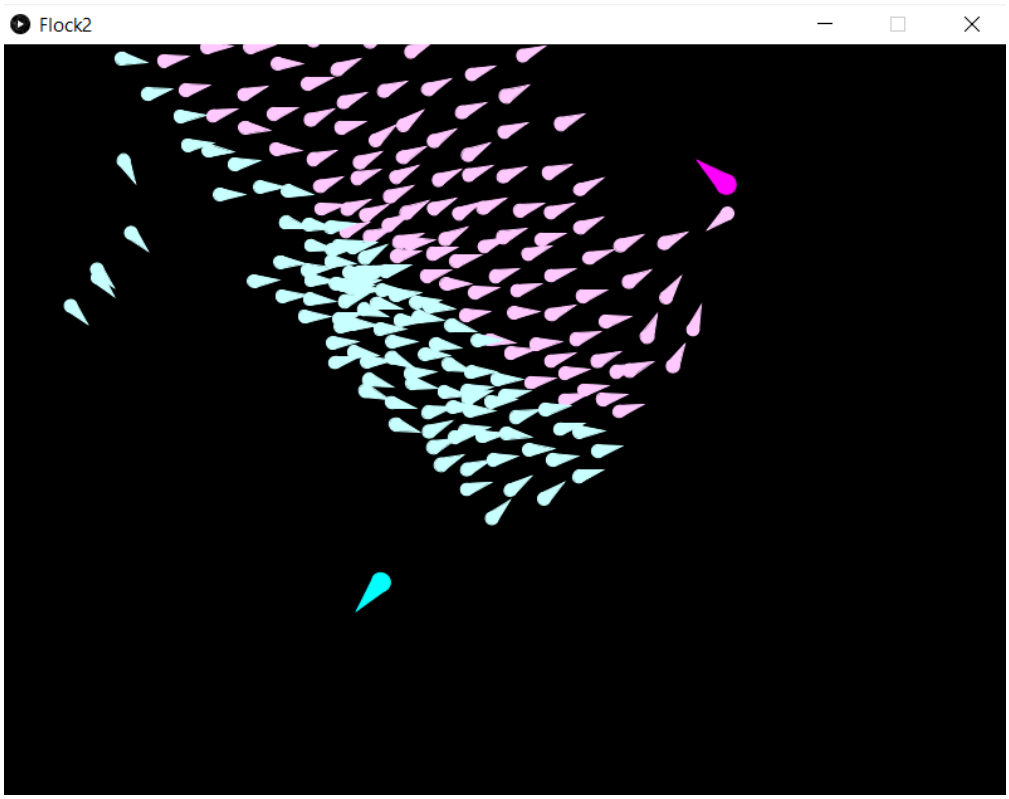


Figure 10: Flocking behavior with two wnaderes

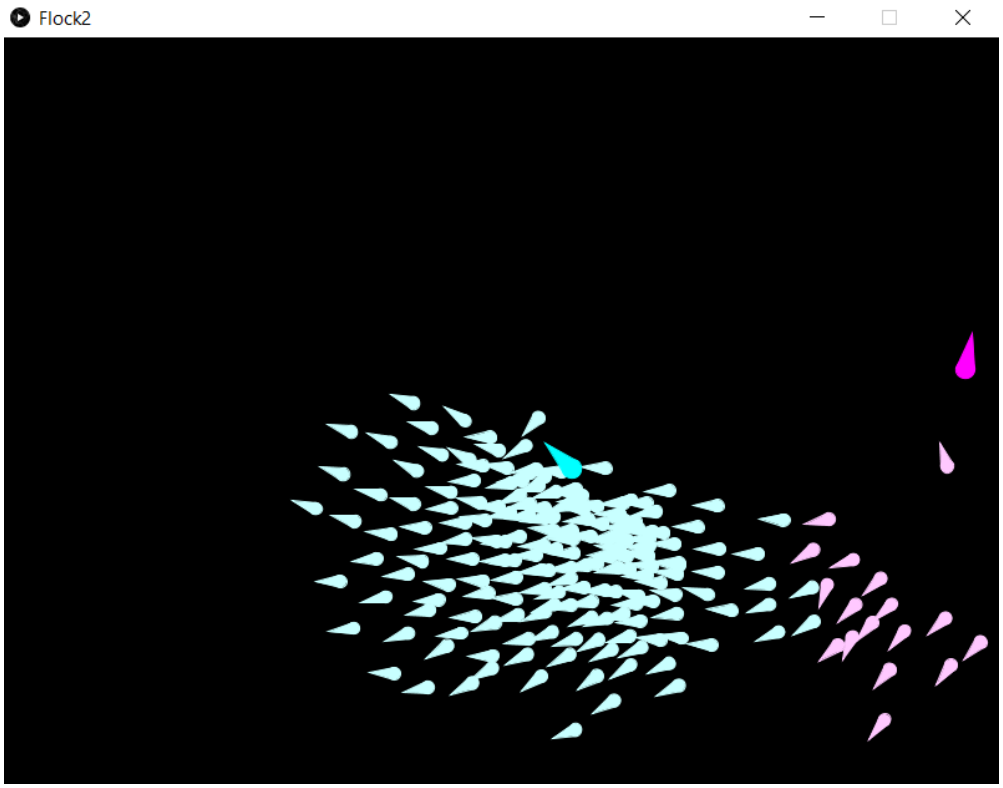


Figure 11: Flocking behavior with two wnaderes