Assignment 1

Movement algorithm analysis

1) Basic-Kinematic motion

Behavior: The character moves along the edges of the screen and comes back to the original spot.

Screenshots:

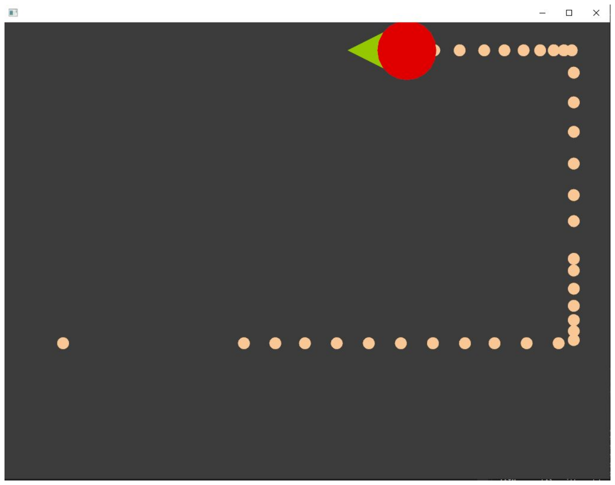

Fig 1.1

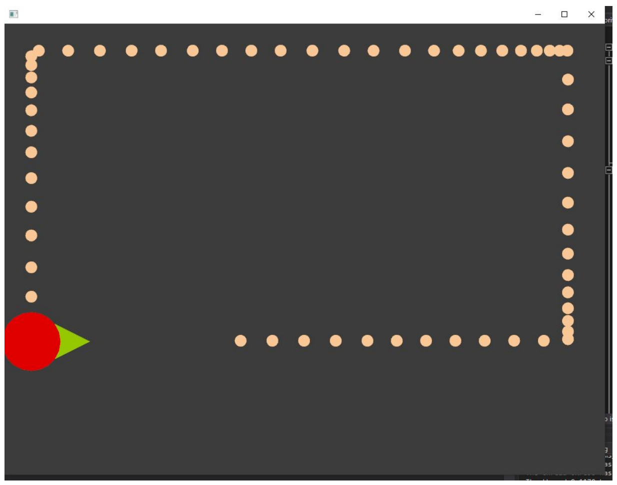

Fig 1.2

The basic kinematic movement involves the fundaments for moving with updating velocity as per acceleration and position as per velocity. In the above figure, we can see the character starts from the left bottom of the screen and completes a rectangle-like circuit to come back to the original spot.

The breadcrumbs(yellow dots) are displaying two things. The route of the traversal and the speed variations. The close-spaced narrow dots show slow speed compared to the wider-spaced dots of increased speed. This tells us that we start with a constant speed ***V*** and accelerate with a constant acceleration ***A*** along time and increase the velocity.

ROHAN PATEL
U1217381

2) Seek steering with arrival
[note: my target is moving with a constant velocity here]

Behavior: The character seeks the target, goes in the direction of the target and reaches the target's specified location.

I have implemented a few different versions of this by tweaking some variables and before we go further to explore them lets get familiar with all the signs in the below-shown picture.

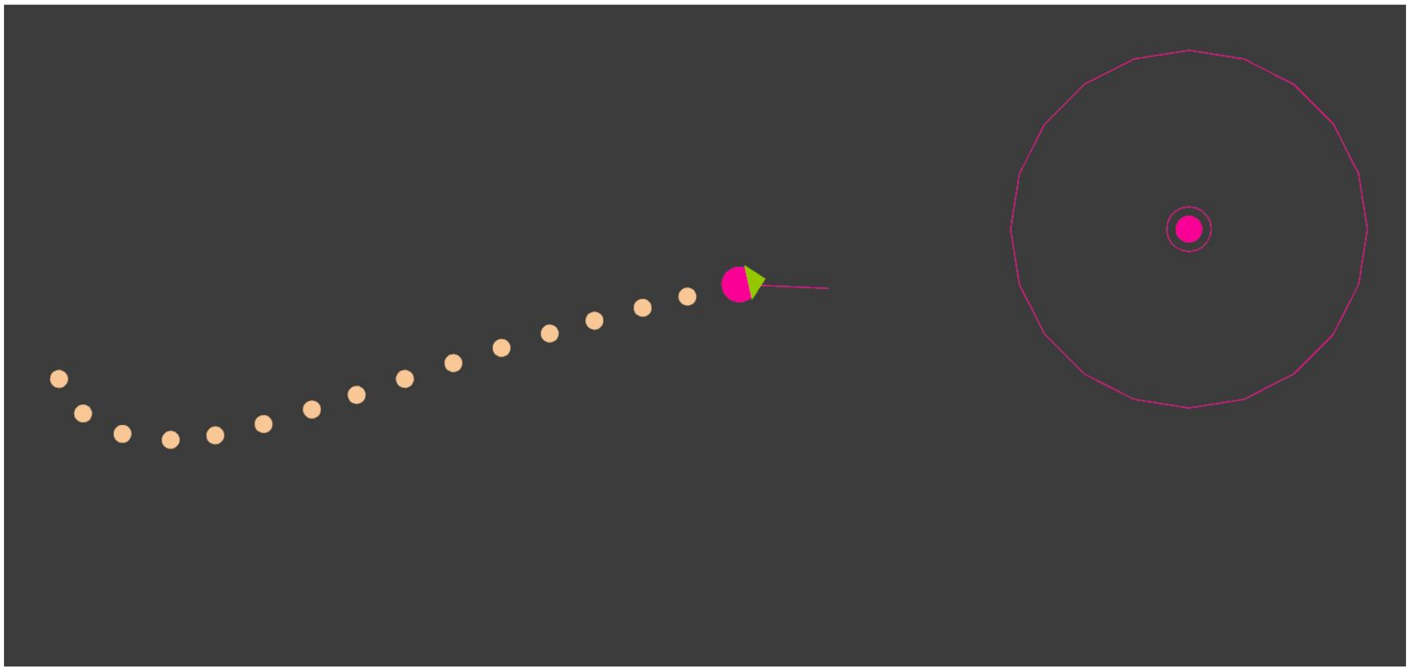

Fig 2.1

The character:

- The **<u>pink circle</u>** with a green triangle shows the character and its direction of velocity.
- The **<u>small pink line</u>** coming out of the character shows the **direction of acceleration**. This will be much helpful in understanding the working of the algorithm.
- **<u>Yellow dots</u>** show the route from where the character has traveled.

The Target:

- The **<u>dark pink circle</u>** inside the two unfilled circles is the character.
- The other two small and big circles are the target radius and slow radius.

I. Acceleration/steering output is in the direction of the target directly

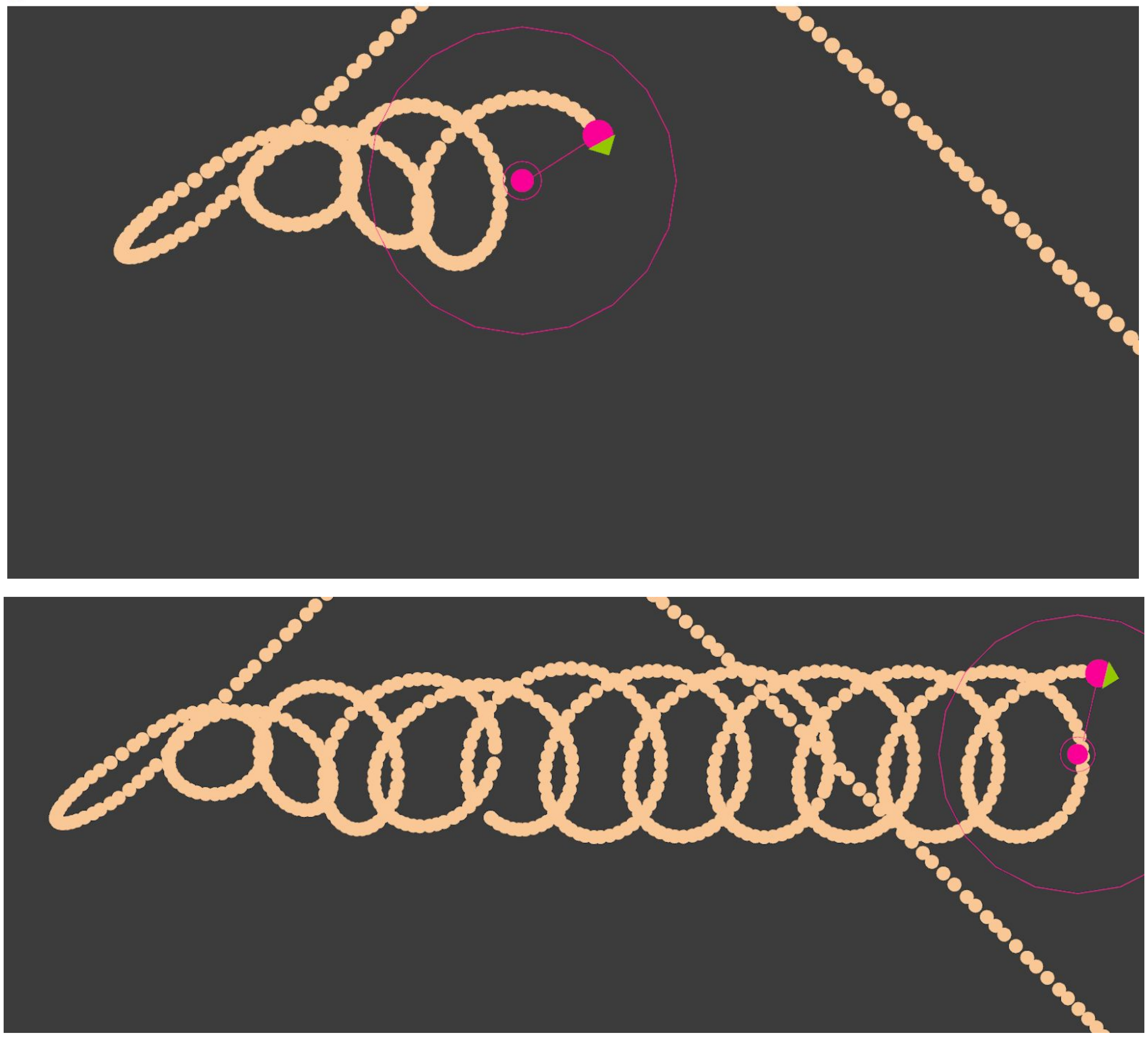

Fig 2.2

In the above figures, we can see the pink line is always in the direction of the target.
This tells us that the steering/acceleration is always in the direction of the target and the target is moving at some velocity.

Here one thing to notice is that it takes the long route to get closer to the slow radius, and once it gets inside the slow radius it tries to reach inside the target radius but it keeps oscillating inside the slow radius and never reaches the target.

II. Acceleration/steering output is the difference between target and character velocity

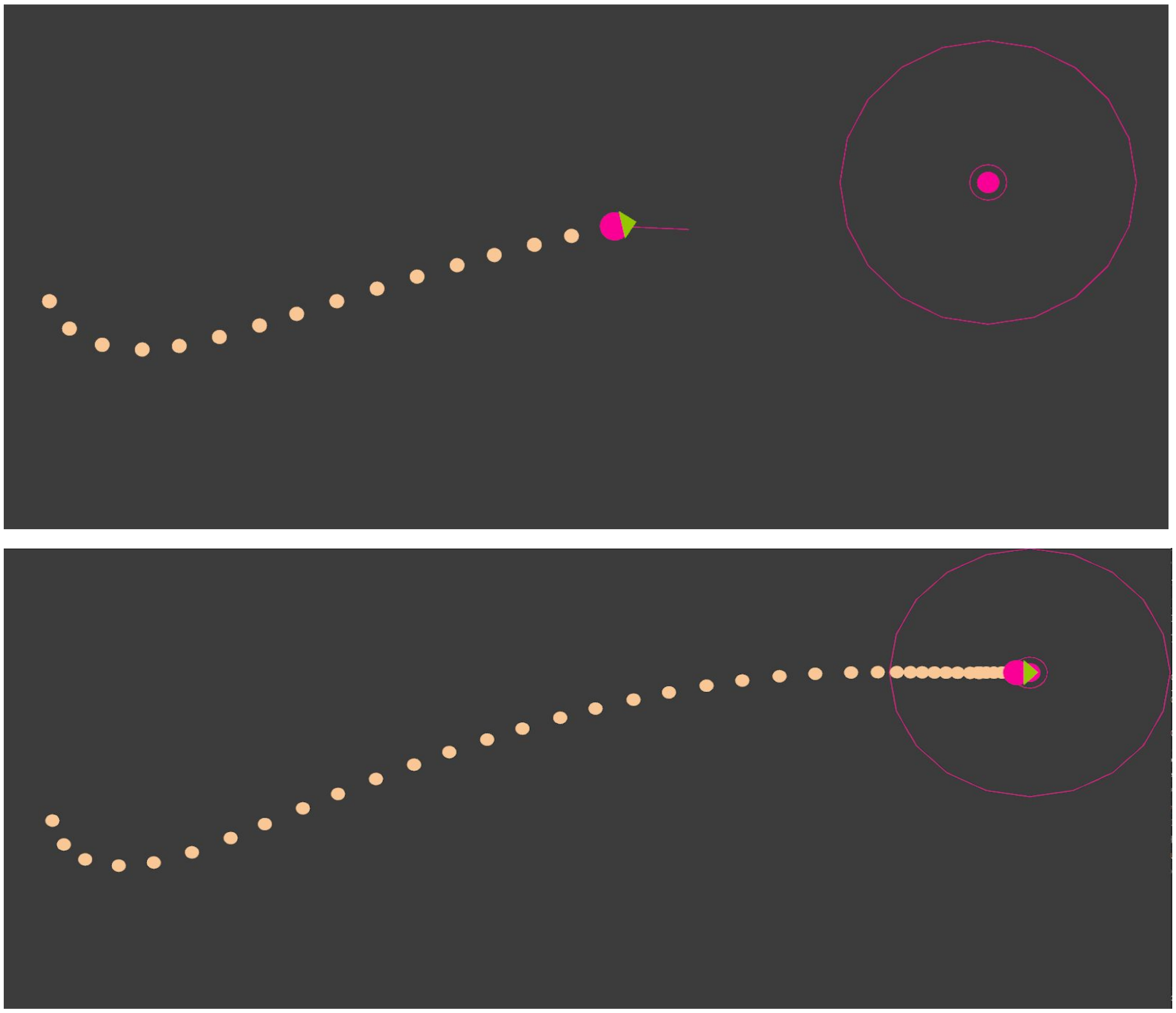

Fig 2.3

Here the steering is adjusted by subtracting the character's velocity from the target's velocity which gives the very interesting direction of acceleration which you can see in the above figure pink line. It is not in the direction of the target as we saw in the previous version. Also as soon as it reaches the slow radius it will try to adjust to the character velocity to reach the target radius by giving the opposite direction of acceleration from the velocity.

This method is the best I have found until now because it uses a very straight route and reaches the target faster.

Some Experiments:

- ❖ **What happens if the slow radius is too SMALL?**

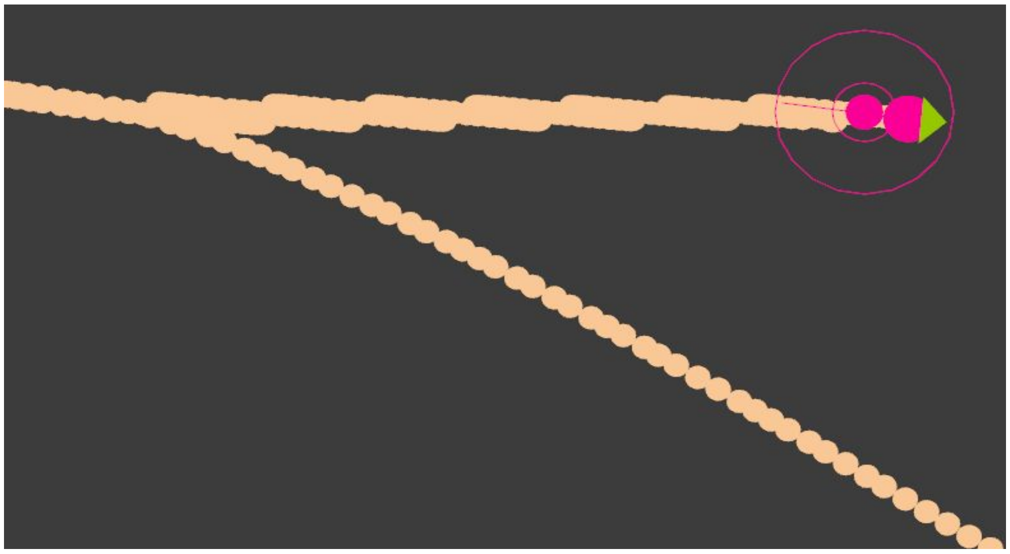

Fig 2.4

In the above figure, we can see that given a very small slow radius it struggles to stop inside the target radius and keeps moving back and forth inside the slow radius but never reaches the target radius.

- **What happens if the slow radius is too BIG?**

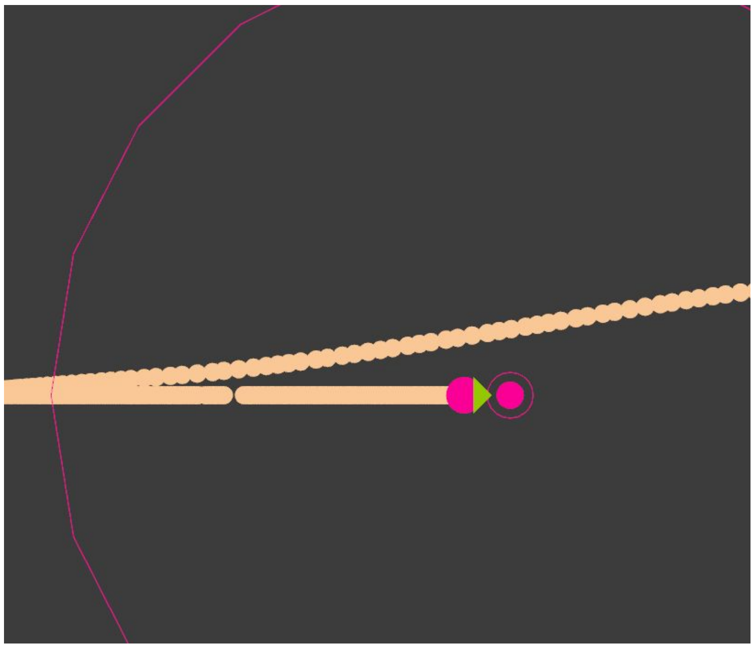

Fig 2.5

In the case of a large slow radius, the character often gets stuck in a deadlock and keeps following the target at a constant rate but never reaches the target radius.

Note: These results might not be the only ones with these parameters and there can be more than one. These are some that I found happening very frequently.

3) Wander steering

Behavior: To keep moving in the direction of the character facing and keep changing the direction randomly as well.

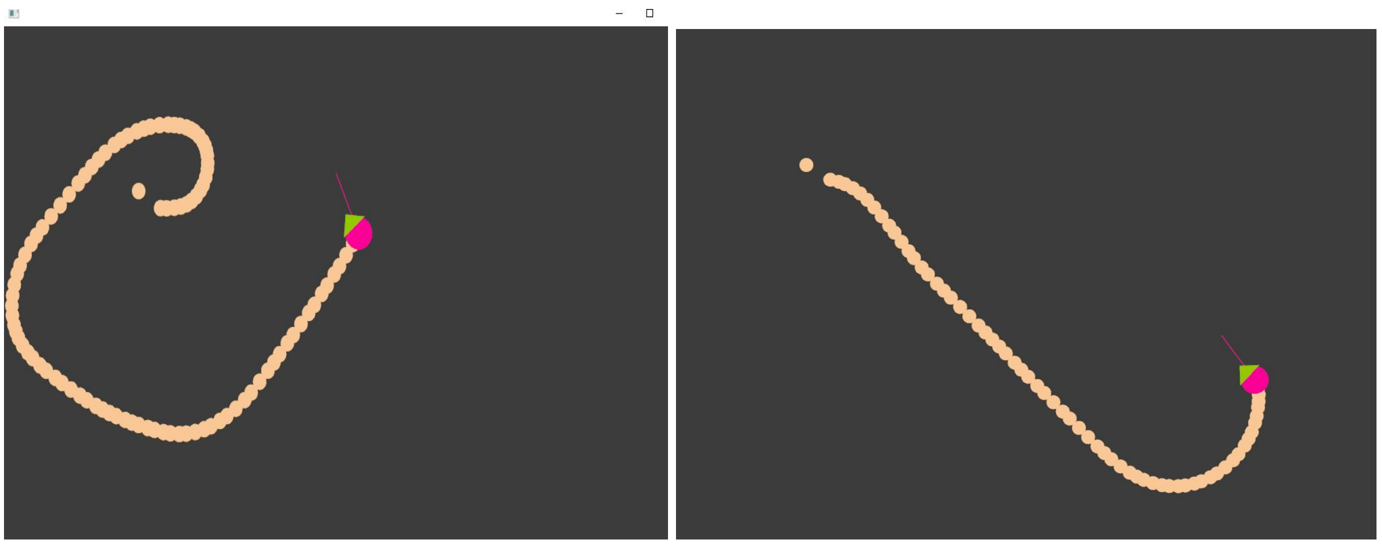
Fig 3.1

In the above picture, the character wanders and changes it’s direction rapidly and move into the direction where it is looking at. Here I tried adding a larger angle threshold and then I was able to achieve the curved path. Before this, the path was mostly linear.

Also, the character changes the direction very rapidly and I add a multiplier to my steering, these are the main factors for changing velocity rapidly as per my understandings.

A problem with the current behavior is that the character keeps changing direction too quickly and to remove this I think I need to add ***lookwhereyougoing()*** to slowly align the direction instead of directly changing it as per the velocity.

4) Flocking

Behavior: Each follower moves in the direction of the leader as if they are following the leader while keeping a minimum distance with the leader and keeping a small distance between each other.

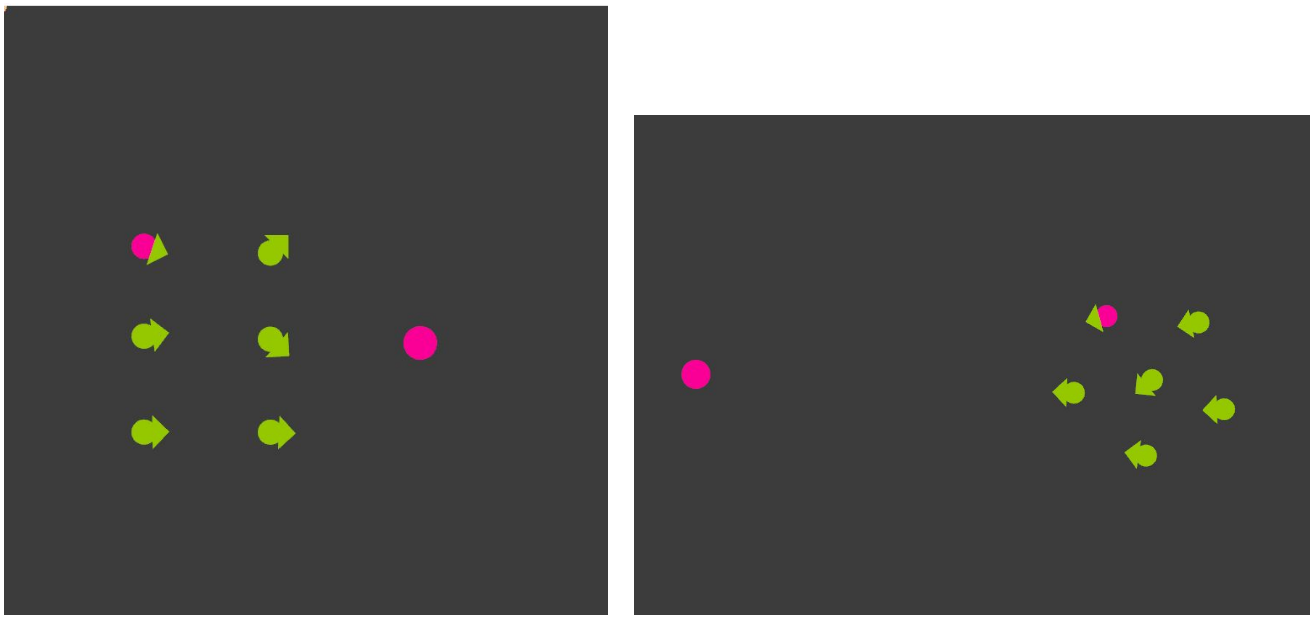

Fig 4.1

I could not get my flocking to work exactly as I wanted. I tried tweaking variables and three crucial components but there are still some things that I am missing and trying to find them.

So in behavior, the flocking starts in a good way but then it starts giving out **two problems**.

1) The followers start moving past the leader sometimes
2) The followers get too close and get grouped at the center of mass

These are the three velocities I applied
1) **A** Seek toward the center of mass
2) **B** Separating when they get inside the target radius
3) **C** Match the velocity with the center of the velocity of all the agents

The cause of problems.

- For problem 1 I think there is not enough counter velocity generated by velocity match and that might be the reason that they move past target exceeding its velocity.
- For problem 2 the separating is not strong enough clearly but when I add severe separation multiplier then it starts separating very aggressively.

While solving this I tried tweaking variables for hours and it produced different results but never the complete flocking of what I desire to be the best. When it satisfies one requirement, it does the opposite for the other. The closest I got is the heard stays around the leader and keeps following it but some of them stay in front of the leader.
Also, I realized that for the direction I need to add align() instead of directly changing the direction.

Summary:
It was a good experience to know how variables and multipliers can do some powerful work if they are balanced properly and can produce great results. The other algorithms work fine except flocking. I am still working on it and trying to get the flocking work.