Alex Kofford

Assignment 1: Movement Algorithms

Kinematic Motion:

I started off figuring out how the basics of OpenFrameworks worked and learned after awhile about the ofTranslate method for drawing. I found that fairly useful for drawing my Boids. Kinematic Motion was pretty straight forward. The main thing I did during this portion of the assignment was setting up my framework for the rest of it. Creating my steering and rigidbody structs as well as the Boid UpdateKinematic method. I decided to create a pool of breadcrumbs so that they didn't get too out of hand. I also decided to have it loop around. So, the Boid will circle forever. (Figure 1)

Seek Steering Behaviors:

For seek steering behaviors I started out implementing seek, with the target just being the center of the screen, once that was working I implemented the ability to click somewhere and have that be the target. Then, not liking the oscillating around the target point I decided to implement Arrive. Arrive only worked slightly better, with it still oscillating even with major increases to the slow radius. (Figure 2) I soon realized this was because once it was inside the target radius the target velocity was set to 0 but without friction or anything we were just sliding through. I found a solution in the book which was that once we calculate the target velocity inside of Arrive we compare it to our current velocity and then set our acceleration based upon the difference. After fixing this Arrive worked wonderfully. (Figure 3)

My next step was to implement changing of my Boid's look direction. I decided it would look best if the Boid faced in its direction of travel. I decided to name my function DynamicLookDir because

Dynamic Looking Where You Are Going seemed too unwieldy. I did it a little backwards, I followed the pseudo code from the book for the LookDir and got it setup and then afterwards I implemented Align. After I got it all setup and I tried testing it sucked really badly. I couldn't find any magic numbers to make it not suck. At least it was rotating somewhat. It took me awhile until I realized I was trying to set the target angle and the slow angle in degrees when I had coded it using radians. Once I fixed it to radians it worked much better.

I feel like in general Arrive is more useful for all the AI I can really think of. Maybe something like a smart missile or an enemy that is trying to run into you? Even then I could see using Arrive to give the player slightly more time to react when it is close or a slight variation on Arrive where you speed up when you get close.

Wander Steering Behaviors:

At this point I felt like I had a lot of the basic algorithms implemented so it was getting easier and easier to implement the new ones. For wander the only algorithm I needed to implement to support it was DynamicFace, which since that derives from Align it was pretty easy. For my wander I made the same mistake with my angles for the look direction so at first it was looking in weird directions. (Figure 4) After fixing that the Wander worked pretty well. I don't really like this type of wandering though, it goes in too straight of a line for too long.

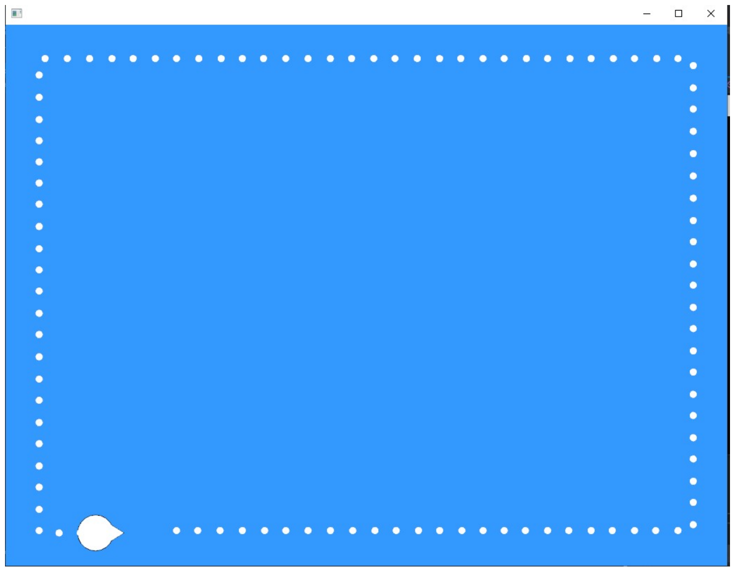
Even changing the parameters around it still has basically the same behavior. I think adding a delay between picking a new angle as well as picking larger angles in general would possibly work better. (Figure 5) I would like it more if it generally stuck inside the screen area. Part of this could be the fact that I made it so when they left the screen it changes their behavior to seeking a random spot inside the screen area. Some people said they made it a wrap around world and maybe the wander would look better in that case.

Flocking Behavior and Blending:

For flocking the only new algorithm I really needed was Separation. I followed the pseudo code from the book and it turns out there was an error. In the book they told you to do target.pos – character.pos to get your direction but this just makes everything run towards each other. Easy fix and it didn’t take too long to figure that out. The other problem I had was when I first implemented it I tried using a decay coefficient of 1, but they weren’t separating at all. After stepping through the code, I realized that due to the math of decay coefficient / distance^2 if I wanted to get a number that was reasonable I needed to ramp that way up. I ended up settling on a decay coefficient of 5000. I think this was in part that the distance calculation was coming from the center of the Boids as opposed to their edges.

After I got Separation all setup I just needed to blend the behaviors. I started off with the suggestion of 1, .8, and .6 for the weights for Separation, Arrive, and VelocityMatch. With this I found the Boids would overlap more than I would like. (Figure 6) It was also around this time I realized it was hard to tell the Boids apart when they overlapped so I gave them a small border. After observing their behavior for awhile I decided I just wanted more Separation so I set the weight up to 3 instead of 1 and got much more desirable results. (Figure 7)

Figure 1:



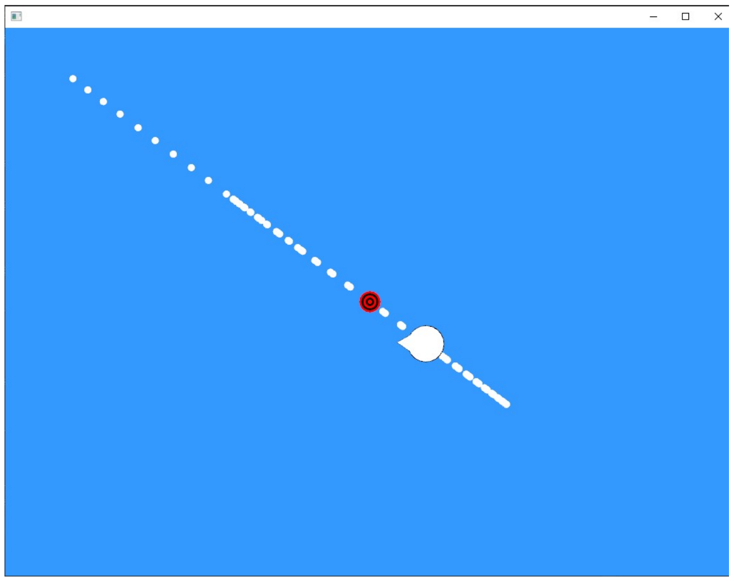


Figure 2

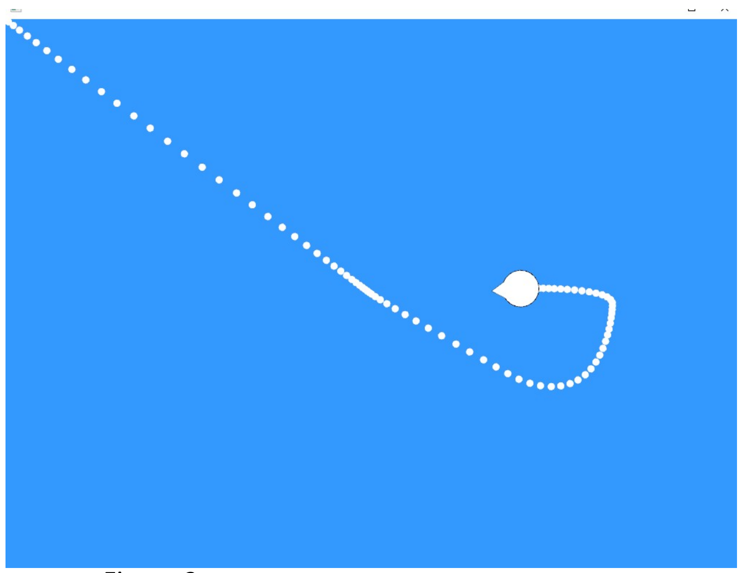
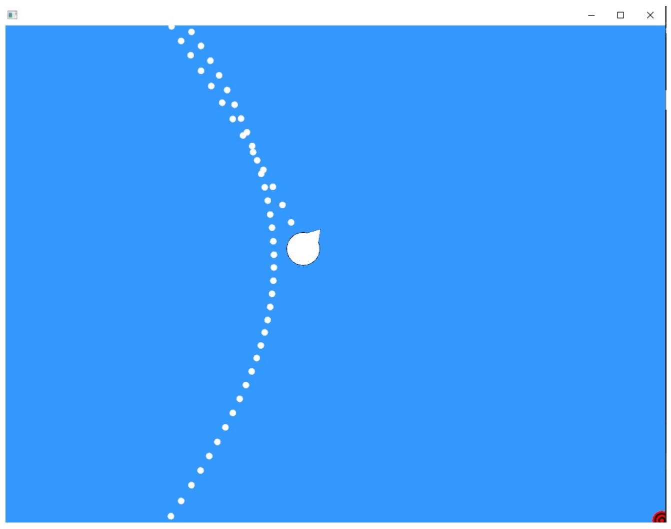
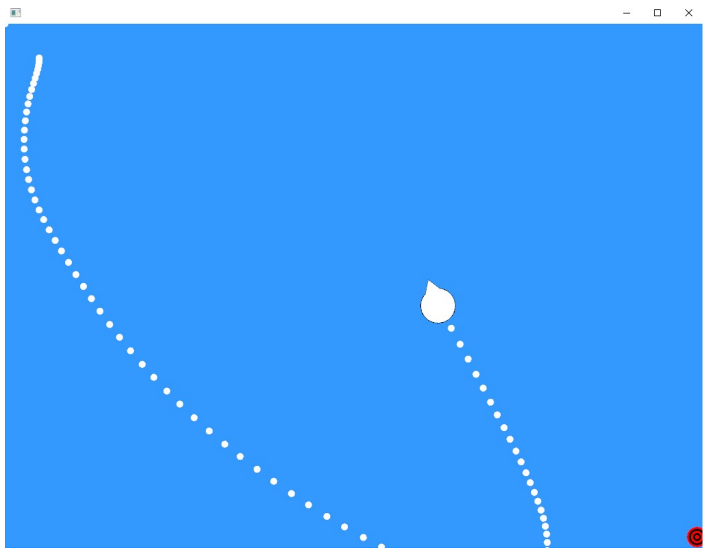


Figure 3





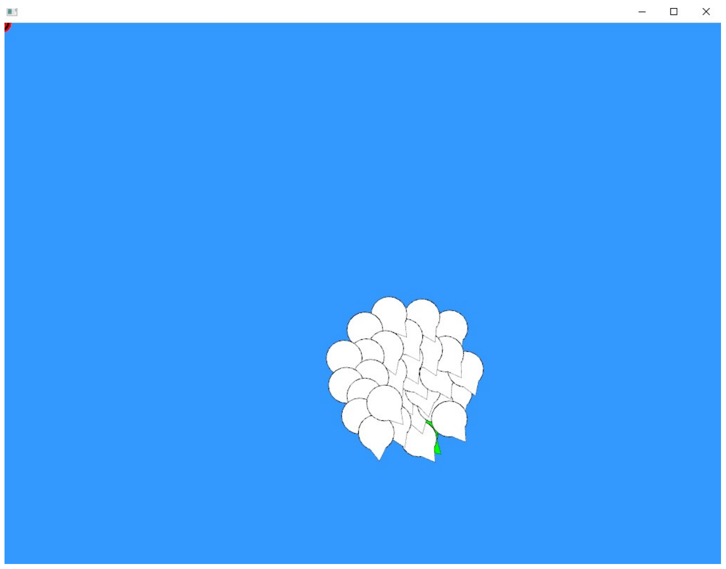


Figure 6

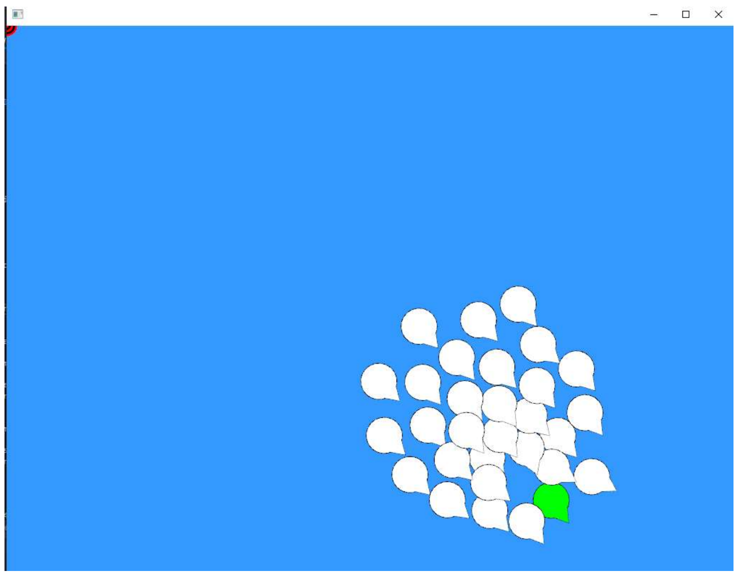


Figure 7