Assignment 1 Process Letter

This assignment was an interesting one for me. On the one hand, I mismanaged my time early on, procrastinating until there was only one week left, leading to stress and missed requirements. On the other hand, I came out of this assignment knowing so much more about movement AI than I did to begin with. While the lectures were helpful and interesting, I am a person who learns best by doing. Having the opportunity to implement these algorithms myself allowed me to really grasp how they worked under the hood.

I started this assignment with setting up and getting familiar with openFrameworks. While I'd never used this framework before, I had experience with SDL and SFML, so the general process was familiar enough that I didn't have too much trouble setting everything up. Thankfully, the openFrameworks website also has lots of helpful documentation and tutorials available for free, so I was able to get acquainted with the basics pretty quickly.

I started with the basics: drawing a boid, moving it kinematically by modifying position directly as a result of velocity, setting up my Rigidbody class, and making the boid move to all four corners of the screen. I had some trouble with updating the boid's velocity, and it would frequently produce unexpected behavior (such as being rendered far beyond its starting position). Luckily, my instructor was able to help me during his office hours, and we resolved the issue rather quickly. I ended up dividing the Update method for my boid into two separate ones, each one being used depending on the type of movement. For kinematic movement, I created a kinematic update function that modified the boid's position directly as a function of velocity (adding velocity to position every frame), and for dynamic movement I utilized the algorithm given to us in class (which changes orientation as a function of rotation and angular velocity, while changing position and velocity based on linear velocity).

After I had gotten the boid to move to all four corners of the screen, I was ready to move on to Seek, my first dynamic behavior. The algorithm itself was pretty simple: get the direction to the target position, normalize that value, then multiply it by some max acceleration value. It was easy enough to understand, but one thing that kept tripping me up was how to orient the boid based on its velocity. I ended up calculating velocity as an angle using the atan2 function, and using that as a rotation. This solution worked perfectly, but I had to sacrifice the use of my angular acceleration value in the SteeringOutput struct.

Once Seek was done and I had figured out how to orient my boids properly, I moved on to Arrive. To start, I duplicated the seek behavior, and added a check every frame to see if the boid was within reasonable distance of the target's position (the target being my mouse clicks in this case). This worked fine, but it wasn't the way we discussed in class, and it made the boid's movement seem pretty unnatural. I then incorporated the use of a "slow radius", where the boid gradually slows its acceleration as it approaches the target. Once another radius, the “target radius” is reached, the boid’s velocity is set to zero, and it comes to a halt, completing the behavior.

I then moved on to Wander. This one was a bit tough for me at first, especially since my notes on this algorithm weren’t very good. I frequently looked to online resources for guidance and understanding, as the overall algorithm didn’t make much sense to me. I didn’t understand why we needed to select a point from a circle in front of the boid–was it not sufficient enough to just select a random vector? After some deep dives into Edirlei Soares de Lima’s slides on steering behaviors and The Coding Train’s various videos on AI behaviors, I was given the answer: the movement is jagged and unnatural. I even implemented it myself and saw how strange it looked. It now made sense to me the need for a point on a circle to be selected, as the differences in angles between points on the circle are far less radical than just a totally random 2D vector.

After that realization it was smooth sailing, at least until I got to implementing Flock. This algorithm was complex and time consuming, and I didn’t understand it quite well at first. Separating it out into three separate sub-behaviors, however, allowed it to really click in my mind. At first I was skeptical–how can these AI perform three behaviors all at once? Edirlei Soares de Lima explains in his slides that these behaviors can be blended together using weights assigned to each of the behaviors. Separation–the behavior that keeps the boids at a certain distance from one another–can be given a weight of 2 for instance, which may be higher than that of cohesion–keeping the boids moving based on a “center of mass”–set to a value of 1.5. The steering values of each of these behaviors are multiplied based on their respective weights, and all three behaviors’ steering results are added together, giving the flocking behavior.

Overall, this assignment was interesting and thought-provoking. The way AI move in games (vehicle movement especially) has always been nebulous to me; there’s always a function in engines like Unreal that handle it for you. To implement these behaviors myself and seeing how they work in real-time was really interesting to me, and it really helped solidify these concepts in my mind. Before this assignment, I only had the lectures to go off of. This is great for an overview of the material, but since I am one who learns primarily by doing, this assignment really helped me understand these behaviors. The seemingly magical properties of steering values (linear and angular acceleration) had all but subsided, as I now understand that velocity is a function of that linear acceleration, which in turn affects the position of an object. I had quite a bit of fun with this assignment, and I learned so much in such a short time!