Process Letter

Kinematic Motion

The way I implemented the kinematic motion is likely to be drastically different from the others. I put my physics static library project that I made last semester into the project and configured to make it work. The rigid body on my Boid is from my physics static library, and thus I have all the features from my library including collision and velocity manipulation working.

Dynamic Steering Behaviors

I implemented two arriving methods, one made from following the video tutorial by The Coding Train from the syllabus, the other designed by myself but inspired from our lecture.

Which arrival method looks better? Why?
The one I designed myself looks much better as it actually stops before reaching the target, while the one following the video tutorial is very likely to pass through the target and then turn back just like the Seek.

Which is more successful? Why?
The one I designed myself is more successful as it utilizes two radii: slow radius and stop radius. It decelerates according to the desired velocity dynamically calculated while within the range between the slow radius and stop radius, and decelerate with 0 as the desired velocity while within the stop radius. The downside of the method from the video tutorial is that it does not use the stop radius, thus its desired velocity is always positive pointing to the target, resulting it not stopping soon enough.

I believe it is much more realistic to have 0 as the desired velocity at some point before reaching the target.

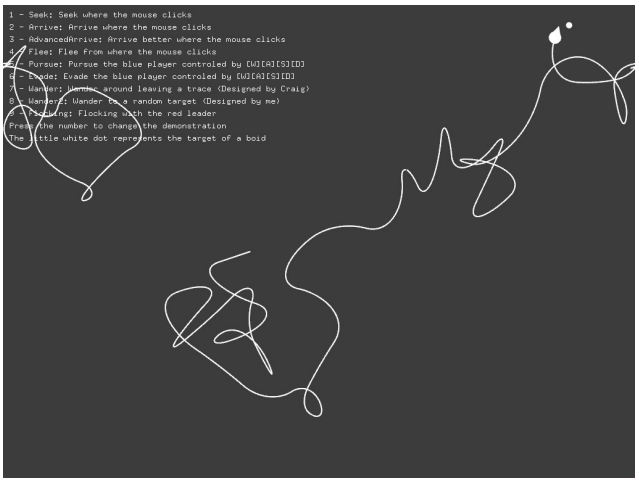
Wander Steering Behaviors

I implemented two wandering methods, one made from following Craig's paper, the other designed by myself.

The one designed by me generates a random point ranging from this.x – wanderRadius, this.y – wanderRadius to this.x + wanderRadius, this.y + wanderRadius. And use the Seek to move towards the target. Whenever the boid reaches within the acceptance radius, it generates a new random target and seek to it.

Wander behavior from Craig's paper:

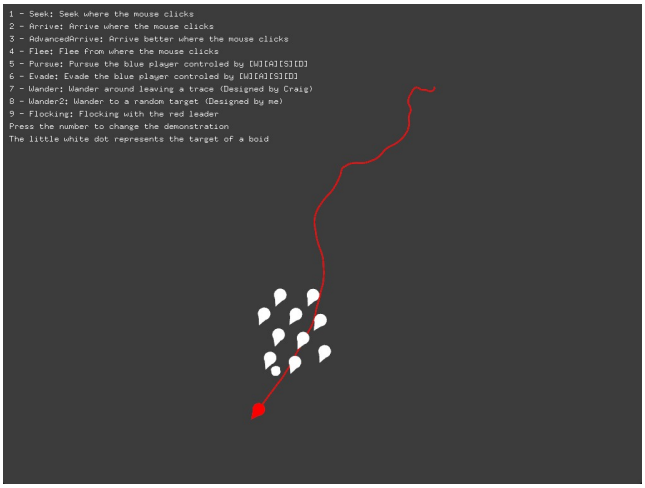
Wander behavior from my design:



Which method for changing wandering orientation looks better? Why?
I believe it depends. If we want the AI to look more like they actually have a destination, then Craig's method looks better as it scarcely make sharp turns. But if we want the AI to look like just pacing without a destination, then I think my method looks better as it is very likely to move back and forth.

For example, if we want to make a citizen in games like Assassin's Creed, we might lean towards Craig's method as a citizen walking on the streets should have a destination somewhere. But if we want to make a guard in games like Assassin's Creed guarding an area, my method should suit better as in that condition the guard should not have any specific destination but rather pacing around randomly.

Flocking Behavior and Blending



During my implementation of the flocking behavior, I observed that the multipliers for align, cohesion, and separation play a heavy role as defining how the flocking behavior look like. If the multiplier for align is too much, the flock might stay still if they are far from the leader and the leader is moving towards the flock. If the cohesion is too much, the flock could easily collide with each other and may even block the leader's way. And if the separation is too much, the boid could easily go backwards in order to avoid collision (see below in the screenshot where some of the boids are pointing backwards).

The more followers, the harder to keep the flock's shape. But I think it is fine as this should be true in real life as well.

It is interesting that by setting a greater weight to the leader, the followers not only followers the leader more, but also focus more on not colliding with the leader than the other followers. This is true in real life, but it is fun how simple it is to implement that in the code.