EECS Project 3 – Steering Behaviors

Project Description and Goal: For our final project, I decided to implement various steering behaviors. Within Unity, I implemented the following steering behaviors: Seek, Flee, Arrival, Wander, Pursue, Evade, and Follow the Leader. The red cylinder represents the agents, and the green cylinder is the goal/target. I have 3 scripts to move the target: 1) Click and drag during runtime, 2) A simple script to move the target in a circular path, and 3) A simple script to move the target in a straight line. These 3 target movement scripts help evaluate the differences between all of these agent steering behaviors. All steering behaviors are scripts written in C++, and attached to the agent game objects in Unity. The movement scripts are attached to the goal cylinder game object in Unity. More information about how to run the project can be found in the readme.txt. The goal of this project is to evaluate the differences, pros, and cons of various steering behaviors.

Evaluation and Results:

Steering Behavior	Visual Evaluation	Pros	Cons
Seek	*Accuracy:* 4/5 *Smoothness:* 4/5	Follows the target very smoothly.	It only *follows* the target, it'll never *catch* it. As for arriving at a target, it'll run into it at a high velocity.
Flee	*Accuracy:* 4/5 *Smoothness:* 5/5	Runs away from the target very smoothly.	If the target is actually trying to catch the agent (pursuit), then fleeing won't work. It flees based off of the current target's position.
Arrival	*Accuracy:* 5/5 *Smoothness:* 5/5	Follows the target very smoothly. When approaching the target, it slows down to a stop. (Eases in).	If the target is trying to run away, it won't *catch* it.
Wander	*Accuracy:* 3/5 *Smoothness:* 3/5	Gives room for some random movement behaviors to account for the target's possible change in route.	Very jittery (could be my implementation though). If the target doesn't change routes, then the steering is only causing a loss of accuracy (randomly moving for no reason).
Pursue	*Accuracy:* 5/5 *Smoothness:* 4/5	Follows and *catches* the target based off its future position. Very Smooth.	We need to tune a parameter: The amount of time we should consider when looking at the target's future position. Looking too far in the future position could cause inaccuracies (due to route changes), looking too close to its current position could cause no pursuit (basically seek).
Evade	*Accuracy:* 5/5 *Smoothness:* 5/5	Runs away from the target based off its future position. Very Smooth.	Same issue as the Pursue steering behavior. See above.
Follow the Leader	*Accuracy:* 4/5 *Smoothness:* 4/5	Basically, seeks towards a leading agent.	Lots of collision between agents. Arrives at high velocity. Never catches targets.

(Note: *Accuracy* refers to how accurately it can catch/approach the target. *Smoothness* refers to the overall smoothness of the motion (such as how fast the velocity is when approaching the target, or general change in velocity).

Conclusion: I found Arrival to be the smoothest steering behavior, as it's able to ease into its target's position. I had to consider the objects width when easing in. As for accuracy, I found Pursue to be the best, since it's able to estimate where the target's future position will be, in order to catch it. The only problem is finding the optimal time-to-look ahead in the target's future position. One solution is to make this time value proportional to the distance away from the target. (As in, don't look too far ahead if the agent is really close to the target). The best possible steering behavior would probably be Pursue + Arrival combined. (To be able to catch the target, but gradually ease in once close to it). Of course, it also depends on what the application is! All steering behaviors have great/acceptable accuracy and smoothness, but some are better than others depending on the application.