Project 4 – Steering and Flocking

CMPS 327

Due Date: Check on Moodle

In this project you will implement several steering and flocking AI behaviors.

Requirements

1. You should create a new scene for each behavior.
2. Name your objects in the scene appropriately to make it obvious which objects are which (e.g. target).
3. Make sure to follow proper organization of your Assets folder. It would be good practice to keep everything in its own folder, like scripts, scenes, materials, etc.
4. Create a new script for each behavior and name it accordingly. In case of fixed target (e.g. seek), your script should allow target to move based on mouse clicks (just like the examples you saw in class).
5. Extreme code documentation is not required, but comments would be nice, especially on anything that isn't instantly straightforward, and major functions.

Tasks

1. You will <u>pick one</u> of three simple waypoint steering behaviors to implement.
    a. **Seek**: The object will calculate a force to move towards an object or point.
    b. **Pursuit**: The object will calculate a force to move towards its target's future position. Pursuit should be used for moving objects.
    c. **Arrive**: The object will calculate a force to move towards an object or point and stop once it has reached its destination. Once the destination is reached, the object should stop

2. You will <u>pick one</u> of three evasive steering behaviors to implement.
    a. **Flee**: The object will calculate a force to move away from a specific point or object.
    b. **Evade**: The object will calculate a force to move away from an object's future position. Evade should be used for moving objects.
    c. **Hide**: The object will calculate a force to move towards a position that places a wall, or some other obstacle, between it and the target. The obstacles may be moving as well.

3. You will <u>pick one</u> of two advanced waypoint steering behaviors to implement.
    a. **Interpose**: The object will calculate a force to move towards a midpoint between two target objects. Two player-controlled objects (targets) with separate controls will be useful here.
    b. **Offset Pursuit**: The object will calculate a force to move towards its target's position but will keep a specified distance from the target's position. This allows for the object to keep within a certain distance of its target

4. You will **implement all** behaviors here, culminating in combining the first three to create a new "flocking" behavior for the fourth. You will create a demo, using this new flocking behavior, adding your own unique aspects.
    a. **Separation**: The objects will maintain at least a minimum distance from other objects within the group.
    b. **Alignment**: The objects will travel in the same direction as other objects within the group.
    c. **Cohesion**: The objects will calculate the center of mass of all other objects within the group, and stay within a certain radius of the center of mass.
    d. **Flocking**: The objects will exhibit all three of the above group behaviors simultaneously. Once you have implemented your basic flocking behavior, play around with it! There are many different combinations of values for the three behaviors' "strengths" that result in interesting flocking behaviors. You could have the flock follow an object exhibiting wander behavior, or have two flocks exhibiting different combinations of the behaviors, with objects bouncing between flocks. There's a large amount of possibilities for you to expand upon the basic idea.

5. This is a **bonus task** (**required for all the students with an honors contract for this course**). You will pick one of three other steering behaviors to implement.
    a. **Wall Avoidance**: The object will calculate a force to avoid colliding with walls by applying a force away from a wall prior to a collision event.
    b. **Path Following**: The object will calculate a force to move to a specified position in a series of positions. Once the object has reached the current position in the list it will then calculate a new force to move to the next position in the list and so on and so forth.
    c. **Wander**: The object will move between two or more points without going in a straight line. This behavior allows for the object to move in what appears to be a random path through the game space.

Hints

1. These behaviors are well documented, both in the resources provided to you, and elsewhere on the internet. Don't try to reinvent the wheel here, especially with the boids algorithm.
2. Instead of trying to fight with Unity's physics engine, do all your movements strictly based on transform math. Take an object's transform and apply all your vector operations to it, instead of trying to use `Rigidbody.AddForce()` correctly. You should also take time into consideration when applying your vector movements.
3. Build early, and build often. Some bugs may arise in your built application that you never would have noticed only running your scenes in the editor!

Deliverables

1. You must submit your code on GitHub (see GitHub Instructions).
2. Submit a zip file on Moodle containing
    a. A README.txt file describing what works and what does not in your application, any known bugs, and any problems you encountered. If you used code from other sources, the readme should also include a list of sources (include a link for each source).
    b. A text file with a link to your GitHub Repository

NOTE: Any Code submitted on Moodle will not be graded. Do not email the code to the instructor or the teaching assistant. Make sure to use the correct version of Unity for this Project.

GitHub Instructions

1. Go to this link: https://classroom.github.com/a/f7xTvP1J
2. After you click on the link, the GitHub page will ask you to sign up for an account or sign into your existing account.
3. The next screen will ask you to accept the assignment.
4. Once you accept, a new repository will be created for you.