194.076 Modeling and Simulation

Sheep Flock Simulation

Tibor Cus, 12325298 *
Aebe Jari van Ravenstein, 12128314†
Leon Olszewski, 11778653 ‡
Alexander Jell, 51820065 §

February 9, 2025
Supervisor: Martin Bicher

This is the report for a group project, in which we aim to simulate a sheep flock as well as herding dogs using an agent-based model. The sheep are modeled as two-dimensional Boids whereas the dogs follow a set of rules chosen by us. Using mainly standard Python and the Pygame library, we implemented the basic model and added some minor features, such as a controllable dog for user interaction or randomly excited sheep for a less predictable sheep flock.

*Prepared the base implementation and animation framework
†Implementation of dog behavior
‡Experimentation with sheep spawning, report and parametrization
§Sheep behavior and controllable dogs

Contents

1 Introduction

Motivation. One of the core functions of a herding dog is to keep livestock together and steer them. By drawing on its inherent authority, rooted in its natural instincts as a hunter, the dog consciously directs the animals' movements. Well-trained herding dogs use sophisticated tactics to guide entire flocks from one point to another in a controlled way.

Traditionally, shepherds used dogs to control flocks of sheep. We argue that sheep flock behavior can be modeled by defining a set of simple rules which each sheep follows.

In this project, we aim to model both of these animals in a shared environment. This should help in getting a better understanding of the behavior of these animals and how the herding process works.

Related Work. This model is inspired by the Boids model which was developed by Craig Reynolds in 1986. Whereas Reynolds simulates "bird-oid objects" in a 3-dimensional space, this work focuses on a 2-dimensional setting. Other than that, no particularly similar work is known to us.

Aim. The goal of this project is to have a working model implementation with the option of parametrization of various variables. Although no quantitative analysis or similar is planned within this project, one can theoretically use the implementation to test which ranges of parameters seem to fit the sheep and dogs.

2 Model

2.1 Modeling

We first describe the conceptual modeling. Hereby we distinguish between the sheep and dogs. Both animals are placed on a two-dimensional plane and their positions are updated simultaneously in discrete time-steps. How their positions are updated is described in the following.

Sheep. For defining the sheep, we relied on the assignment description provided by the course instructor, Dr.techn. Martin Bicher. Each sheep has two main properties that can be formulated as two-dimensional vectors: a position and a velocity. In each time iteration, each sheep's velocity consists of a weighted sum of the following three vectors:

A "cohesion" vector pointing into the average direction of all neighbors. d_o is an observation radius, so a distance how far a sheep can see neighbors.

$$\vec{w}_1 := \frac{1}{|\{j : |\vec{x}_i(t) - \vec{x}_j(t)| \leq d_o\}|} \left(\sum_{j:|\vec{x}_i(t)-\vec{x}_j(t)|\leq d_o} \vec{x}_j(t) \right) - \vec{x}_i(t).$$

An "alignment" vector consisting of the average velocity of all neighbors.

$$\vec{w}_2 := \frac{1}{|\{j : |\vec{x}_i(t) - \vec{x}_j(t)| \leq d_o\}|} \left(\sum_{j:|\vec{x}_i(t)-\vec{x}_j(t)|\leq d_o} \vec{v}_j(t) \right).$$

A "separation" vector pointing into the negative direction of sheep within a collision distance d_c. This ensures that sheep do not collide into another.

$$\vec{w}_3 := -\frac{1}{|\{j : |\vec{x}_i(t) - \vec{x}_j(t)| \leq d_c\}|} \left(\sum_{j:|\vec{x}_i(t)-\vec{x}_j(t)|\leq d_c} \vec{x}_j(t) - \vec{x}_i(t) \right).$$

The new velocity is then a weighing of the above vectors. In addition, we ensure the sheep can never run faster than a certain v_{max}. Also, to realistically simulate that sheep cannot or do not want to move for long periods of time as birds or small fish might, we introduce a damping factor δ. It leads to sheep coming to a halt if no external force (such as a dog) is introduced.

$$\vec{v}_{raw} = \vec{v}(t)l_0 + \vec{w}_1(t)l_1 + \vec{w}_2(t)l_2 + \vec{w}_3(t)l_3,$$
$$\vec{v}(t+1) := \min\left(\frac{v_{max}}{|\vec{v}_{raw}|}, \delta\right)\vec{v}_{raw}.$$

We also add a fear speedup factor for the sheep that causes the sheep to move faster the closer they are to the closest dog.

$$\text{speedup}_{fear} = \begin{cases} 1, & \text{if } d \geq r \\ f_{\max} * \left(1 - \frac{d_{closest}}{d_o}\right), & \text{if } d < r \end{cases}$$

- d_{closest} is the distance to the closest dog,
- d_o is the dog observation radius,
- $f_{\max}$ is the maximum speedup reachable through fear

We then change the speed-limiting function to

$$\vec{v}(t+1) := \min\left(\frac{v_{\max} * speedup_{fear}}{|\vec{v}_{\text{raw}}|}, \delta\right)\vec{v}_{\text{raw}}.$$

Dogs. The Dogs, like the sheep, have a "separation" vector (see formula above). They try to avoid colliding directly with sheep and other dogs. In addition, they observe the center of mass of the sheep within their observation radius.

$$\mathbf{v}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{p}_i - \mathbf{p}_{\text{self}})$$

- $\mathbf{p}_i$ represents the position of each neighbor.
- $\mathbf{p}_{\text{self}}$ is the position of the agent/dog.
- $\mathbf{v}_{\text{avg}}$ is the average distance vector.
- N is the number of close neighbors.

They will try to move mostly perpendicularly to this vector. There are 2 different weights, one for the movement along the vector and one for the perpendicular movement. In this way, we can alter how close the dogs can get to the sheep before the sheep avoidance vector overpowers the vector pointing towards the flock.

Lastly, one dog is tasked with finding and catching strays. This works according to the process detailed below.

Step 1: Compute the Average Position of Neighbors

$$\mathbf{p}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i$$

Step 2: Compute Maximum Distance and Total Displacement

$$D_i = \|\mathbf{p}_i - \mathbf{p}_{\text{avg}}\|$$

$$D_{\max} = \max_i D_i$$

$$D_{total} = \sum_{i=1}^{N} D_i$$

Step 3: Compute the Push Position for the Stray Neighbor

$$\mathbf{p}_{\text{push}} = \mathbf{p}_i + 0.1(\mathbf{p}_i - \mathbf{p}_{\text{avg}})$$

Step 4: Apply Steering Logic

$$\text{threshold} = 10 \times \frac{D_{total}}{N} + 0.2$$

$$\mathbf{v}_{\mathbf{steer}} = \begin{cases} \mathbf{p}_{\text{push}} - \mathbf{p}_{\text{self}}, & \text{if } D_{\max} > \text{threshold} \\ (0,0) & \text{otherwise} \end{cases}$$

- $\mathbf{p}_i$ represents the position of each neighbor.
- $\mathbf{p}_{\text{avg}}$ is the average position of all neighbors.
- D_i is the magnitude of the difference between a neighbor's position and $\mathbf{p}_{\text{avg}}$.
- D_{max} is the maximum of these distances.
- D_{total} accumulates all these distances.
- $\mathbf{p}_{\text{push}}$ is the pushing position from which the dog will push the sheep.
- $\mathbf{p}_{\text{self}}$ is the position of the agent/dog.
- $\mathbf{v}_{\mathbf{steer}}$ is the steering vector applied to the dog.

This will cause the dog to find the sheep that is furthest away from the flock and will get behind it to push it back to the flock. If the furthest away sheep is within the threshold, the dog will assume normal behavior.

2.2 Implementation

We implemented the model using Python and the *Pygame* library. We also made use of *random* and *numpy* as helper libraries for introducing (pseudo)randomness and to easier find neighbors and calculate distances.

The following classes are used in our code:

- **Environment:** Represents the grassy area where the sheep and dogs coexist. Includes attributes such as height, width, number of animals and methods for initialization.
- **Animal:** A parent class for dogs and sheep.
- **Sheep:** A sheep class. It implements all the attributes and methods as defined in section 2.1.
- **Dog:** A dog class.
- **ControllableDog:** A child of the dog class. This dog can be controlled by a user using the arrow keys or W, A, S, D on the keyboard.

Many aspects of our implementation depend on parameters which are set in a .env file. For instance, the number of sheep and dogs, the weights for the velocity vectors of sheep or an animal observation radius are set here. For many of these parameters, we had to repeatedly run simulations and adjust according to how we saw fit.

Our implementation offers two ways to spawn sheep: normally around the center and uniformly random. Moreover, with a low probability p, sheep start becoming "excited" an running in random directions, ignoring the rules defined in section 2.1. This was to test the "robustness" of the flock".

3 Simulation Results

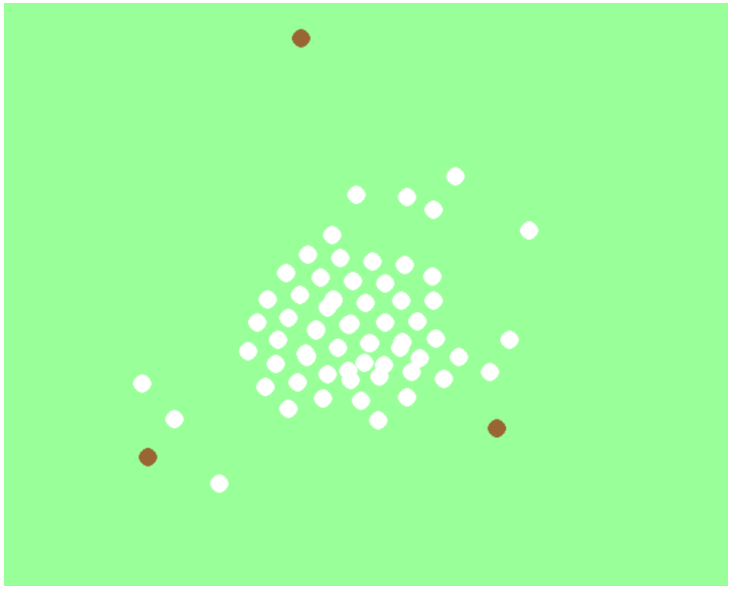
In general, the simulations run in a fashion that we are satisfied with. As mentioned, we did not intend to run systematic experiments with certain performance metrics. Moreover, validation using real data is difficult as barely any data exists on this subject. Instead, the focus here was to create a "realistic" simulation of this traditional herding process. In order to visually validate the model, one can look at video content (example).

Whilst our simulation is best observed when running it live as it captures the dynamics of the process, we included screenshots of runs. The background is green to represent a grassy area, the sheep are colored white and the dogs are brown.

The following image is a normal sheep spawning without any dogs after approximately 3 seconds. We can see that due to the strong cohesion force, a flock quickly starts to form.

The following shows sheep spawning uniformly, also after approximately 3 seconds. We can see that the flock merging process takes longer here because of multiple clusters forming at first.

Here is a sheep flock where 3 dogs circle the flock and push it inwards. Particularly, the dog in the bottom left corner has 3 sheep targets that they want to merge into the rest of the sheep.



4 Discussion

Conclusion. The boids formulas keep the sheep together and we generally had to make it harder for the dogs to make the herding challenging. We added additional behavior, such as random movement for certain sheep or proximity speed-up, to punish incorrect herding. Initially, this task also sounded like a classic task to use reinforcement learning to train dogs on herding, but simple strategies were enough to keep the sheep together.

Outlook. The simulation could be extended to give the dogs additional tasks. One scenario could be to vastly increase the area and define a target spot toward which the sheep should be herded. Herding the sheep in a spot is rather easy as a result of the natural behavior of the boids to group up and stick together. It would be really interesting to create a scenario or formulate a challenge for the dogs that is hard enough to incentivize reinforcement learning to train the dogs to herd.