Assignment 4 Documentation

Architecture

During the networking assignment, we added functionality to divide physics processing for the various non-player entities between all of the players connected to the game. We will continue to leverage this system to ensure that AI processing is only done for entities which the client has control over, though making AI work in multiplayer is not among our top priorities for this project. Since the AI agents in our game are simple, we will use state machines and/or fuzzy logic to determine what the AI should do. Our proposed AI behaviors are, for sheep, to wander, flee from the goals, and attack players, and for chickens, to flock, to scatter, and to hide. We believe these behaviors will help the player view these two entities as distinct, rather than simply two, differently shaped, soccer balls.

Expected Behaviors

Behavior	**Description**
Sheep::wander	Default state. The sheep wanders slowly and aimlessly around the world.
Sheep::flee	Triggered when near a goal. The sheep actively moves away from the goal.
Sheep::attack	May be triggered when near a player. The sheep attempts to ram the player, knocking them back and temporarily stunning them.
Chicken::flock	Triggered from the hide state when another chicken moves nearby.
Chicken::scatter	Triggered when the chicken is near a player or a goal. The chicken attempts to move away from all entities, including other chickens.
Chicken::hide	Triggered after the scatter state. The chicken attempts to find a spot in the world that is not visible by any player. Suggested implementation: compute a "visibility gradient," then perform a gradient descent. Only attempting to hide from the nearest player may be sufficient for this behavior.

Division of Labor

Task	**Assignee**	**Status**

Folder structure cleanup	JB	Complete
Convert animals to KinematicBody, system for animals to specify their velocity vector	Arik	Complete
Sheep wander state	Arik	
Sheep flee state	JB	Work in progress
Sheep attack player state	Arik	
Chicken flocking behavior	Skyler	
Chicken scatter behavior	Skyler	
Chicken hide behavior	JB	

Stretch Goals

Task	**Status**
Correctly replicate AI systems over the network	Bugs
Implement animal "personalities" using fuzzy logic, perhaps using animal coloration to hint to players how the animal might react	Not started

Flock Design

Implementing the believable flocking of chickens is a non-trivial task. We will opt to use the boids approach for implementing this behavior, as it is a tried and tested method mimicking flocks of birds, fish, and other herd animals. The implementation of boids includes the following traits for each chicken:

1. Separation - Steer to avoid crashing into nearby chickens
2. Alignment - Steer in the average direction as nearby chickens
3. Cohesion - Steer towards the average position of nearby chickens

These three properties alone will create bird-like flocking behavior. Implementing these will not be too involved, as we simply need to access the other chickens sufficiently close to a cutoff (perhaps not directly behind the current chicken) and to create an appropriate rotation for our velocity vector. However, implementing this in three dimensions will require two possible considerations.

1. Efficiency. The three characteristics require each chicken to check the location of all other chickens. The occasional call would not be cause for concern, but it's unclear if

running this check in every physics tick can cause a noticeable performance hit. If so, a spatial partitioning system may need to be implemented to solve this, where each chicken is given only a collection of chickens from nearby spatial cells. However, this setup may take more operations at smaller scales our game operates at since there are a limited number of chickens, so it remains to be seen if this will actually be an improvement.

   a. If such a partioning is needed, a separate "chicken manager" would make sense, as there would be non-trivial code to organize these cells. If there is not a performance hit, then the game manager can simply give each chicken the location of all other chickens
   b. Alternatively, I believe simply multithreading these checks for each chicken would be the most effective solution. We would spawn a thread for each chicken's check, though I'm not sure how to guarantee execution of these tasks properly per physics tick if this is possible.

2. Obstacles. Chickens should steer around physical objects in our world. This will require raycasts out in our world to find nearby objects, which means deriving an efficient system for casting enough in the world to find a good path around these objects. There are two approaches:
   a. Steer in the opposite direction of the obstacle. This just requires a single raycast in the orientation vector of our chicken, from which we can flip our velocity if an object is sufficiently close.
   b. Cast many rays in a sphere. This would be an involved process that allows us to find a "close enough" vector angled from the forward vector that the chicken can turn to. Digging around online, it seems a good bet to do this would be:
      i. Generate a collection of points on a line. The iteration should start at the center, but then alternate between a new farthest point on the left, then on the right, repeating until done. This way we can consider both directions (moving left or right) equally.
      ii. Project these points on the edge of a circle, so that it acts as possible vectors our chicken can move in. We then need to rotate this collection to the normal of the ground the chicken is on to consider its perspective.
      iii. We then iterate through these vectors as ray casts until one is found without an obstruction (ignoring the ground). Since these vectors are originally points generated from the center, this will find the first vector closest to the bird's "look" direction to steer to avoid obstacles, which is the exact behavior we'd like.
      iv. We will then need to keep a parameterized constant to turn our bird towards this new vector for the next tick.
         1. The good news is that we could reuse this behavior for scattering as well.