Shepherd, James WVRV402 12 July 2021
217257968 Assignment3

The following is a report of my implementation for assignment 3.
Having completed assignment 1, I chose assignment 3 over assignment 2 mainly because I didn't have much time left to complete it and I didn't feel comfortable trying to learn AR implementations in such a short period. Assignment 3 held the potential for using pre-built assets and packages which would save a lot of time, plus it only relied on standard unity features which I was more familiar with (rather than AR).
On top of that, I didn't really have any great ideas for assignment 2, however assignment 3 felt more diverse and exciting. I could get creative and have some fun with it.

Unfortunately this assignment was more challenging than I had hoped and I was not able to implement all of the required functionality. I implemented as much as I could however I wasn't able to implement the oct-tree or the pack hunting features for this assignment.

I decided to focus on an ocean scene for this assignment. I included a variety of plant life, some rocks (for obstacles), fish which flock/school together, ground fish which wander the seabed (not flocking) and a few sharks which wander around.
The fish both feed on the plants and the sharks feed on the flocking fish.

Almost all of the algorithms I used were found online although many of them needed quite a lot of editing to get them to work for this project. I mostly used the online resources for reference and as a starting point.
I made use of unity's built in animator controller system to manage the objects' state changes as it is essentially just a state machine. I made just two animator controllers, one for animals and one for plants.

All living entities have a lifecycle, switching from 'Baby' state through to 'Elder' and then 'Dead'. Objects will remain in a particular state for a fixed amount of time before progressing to the next stage. Most of the growth occurs during the 'Baby' stage.The animals have been set to switch color as they get older. Babies are a lighter colour, adults are brighter and older animals are a darker colour.
All animals also have a hunger level and a sex drive (because I didn't know what else to call it). The hunger level takes priority but when either of these values reaches a certain limit the animal will switch states.

I didn't want to waste time placing all of my plants and rocks into my environment so I built a simple terrain and made use of a vegetation spawner that I found on the asset store to automatically fill my terrain. The problem with this, I discovered, is that the objects added to the terrain weren't considered normal game objects and so none of the functionality would run. So to fix this I found another online resource which scans a terrain and replaces all of the terrain objects with actual game objects.

The plants have a simple life cycle. They start small and grow bigger (mostly in their early stage). Once they reach ‘adulthood’ they start spawning periodically. This simulates spreading seeds. The probability of spawning and number of seeds spawned are all randomized. The spawned seeds then follow the same lifecycle.
If a plant is eaten by a fish it reduces in size. If it gets too small then it dies.

The ground fish make use of a navmesh to move around. They are set to avoid rocks but not plants. This is because fish often swim through plants and seemed more realistic to me. Generally, they simply wander around. To achieve this I found a random point within a given radius, mapped that point onto the navmesh and set that point as the fish’s destination. Once the fish reached that point it would find another random point.
When the fish gets hungry it searches the nearby area for plants. It chooses a plant at random and sets that as its new destination. When it gets there it starts eating. It will eat until the plant is finished (and then find another plant) or until it is no longer hungry.
The fish follow a similar process when mating. Once they reach their prospective mate they attempt to mate. These attempts have a random success probability although they will fail if the prospective mate is already mating or is fleeing.
The ground fish hide from sharks. To do this the algorithm scans the navmesh for edge points, calculates the distance of each point from the enemy, determines if the point is a safe distance away and is hidden from the enemy, and if so then the fish goes there.

The flocking fish make use of a prebuilt boid flocking algorithm that I altered.
The fish all belong to a flock (I could have several flocks), the flockManager moves to the flock’s target point around within the area so that the flock is following a moving target. The boidManager keeps track of the average information of all boids. Each boid uses this general information to stick together in the flock.
These fish flee from the sharks instead of hiding. When sharks are near, the fish simply calculate the average direction away from all nearby sharks and move in that direction.
These fish follow the same feeding and mating processes as the other fish.

The sharks also wander around. They use a similar method to the ground fish however they obviously don’t map their destination point onto the navmesh. They are also set to gradually turn because sharks can’t turn as fast as small fish.
When the sharks are hungry they continue wandering until a fish gets close enough, then the shark quickly speeds up towards the fish but just for a short burst. It then slows down again. If the shark manages to eat the fish then the fish is destroyed and the shark is less hungry.
The mating process is the same as the other fish.