CMPSC 383: Multi-Agent and Robotic Systems

Final Project Progress Report

1 Boids

Boids is a simple simulation of flocking behaviour which mimics the appearance of a flock of birds [1].

2 Language Choice

We chose Haskell as the implementation language for our Boids simulation. Haskell is a purely functional language with lazy evaluation. It has been observed that Haskell supports the rapid prototyping of software systems, allowing working systems to be implemented quickly and with minimal complexity [2]. This observation, along with our previous experience in the language, influenced our choice of Haskell as a platform for our project.

As our class experience with implementing Multi-Agent simulations has been solely through the object-oriented programming paradigm, our language choice presented a challenge, but also a valuable learning experience. Our implementation of the Boid agents involves the definition of an abstract data type, which contains the position, velocity, and neighborhood radius of an individual boid. In Haskell, this is defined as follows:

```
data Boid = Boid { position :: Point
                 , velocity :: Vector
                 , radius   :: Float
                 }
```

Note that Haskell does not allow mutation of existing values. Therefore, once a Boid is created, it cannot be mutated, and instead updating the BoidâĂŹs state requires the creation of an entirely new Boid instance.

Also, in contrast to a corresponding OOP implementation, which might define some Boid behaviour to accompany this basic data structure, this Boid data type is kept distinct from its update method. Instead, we define a type called `Update`:

```
type Update = Boid -> Boid
```

Thus, a function of type `Update` is a function that takes a `Boid` and returns a new `Boid`. We use `Update` to define `Behaviour`:

```
type Perception = [Boid]
type Behaviour = Perception -> Update
```

This defines a `Behaviour` as a function that maps a `Boid`'s neighborhood to an update function. These examples demonstrate how implementing a multi-agent simulation in a purely-functional language requires a different conception of what it means programmatically for an Agent to behave.

3 References

[1] Christopher Hartman and Bedrich Benes. Autonomous boids. *Computer Animation and Virtual Worlds*, 17(3-4):199–206, 2006.

[2] Paul Hudak and Mark P Jones. Haskell vs. ada vs. c++ vs. awk vs.... an experiment in software prototyping productivity. *Contract*, 14(92-C):0153, 1994.