Semester Project OOAD

Interim Project Report

Individual: Dieu My Nguyen

Project title: Agent-based model of firefly mating strategy

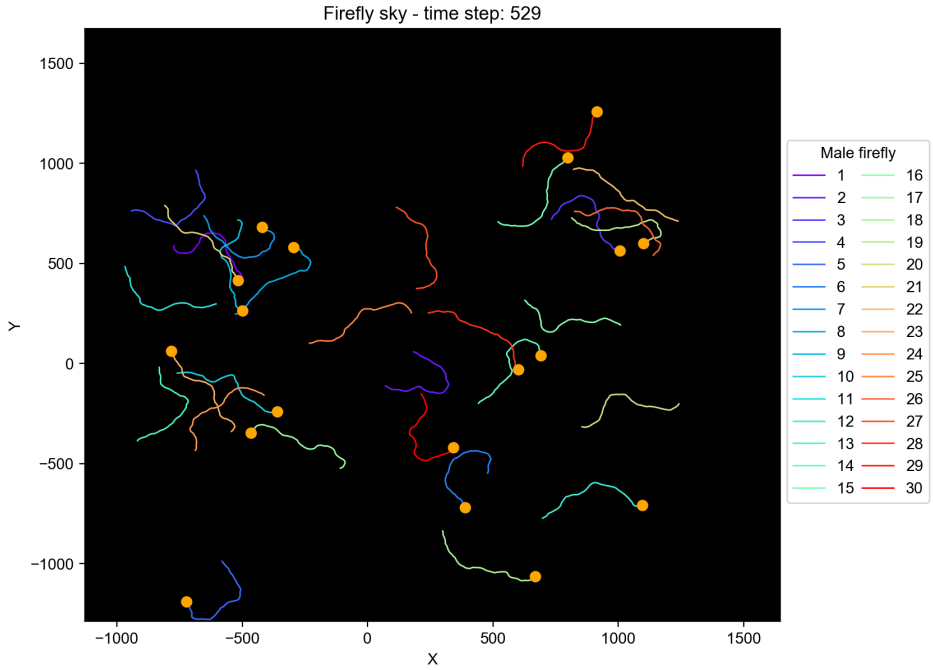
Summary

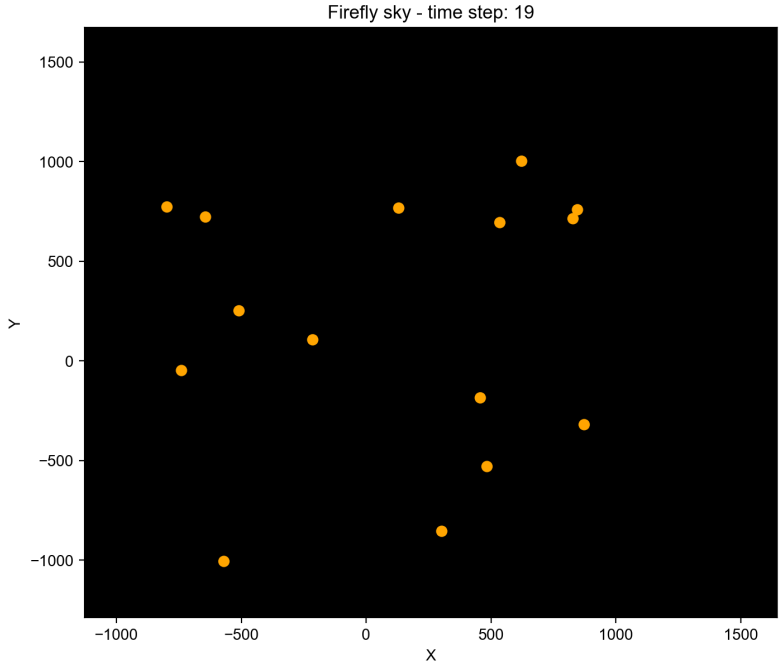
What's done:

So far, I have implemented a core theoretical component of the agent-based model: The correlated random walk model for firefly motion and signaling strategy via intermitten flashes. Since this model has an important theoretical underpinning, I spent some time to understand the random walk model. It is fairly simple: At each time step, a given male firefly picks a random turning angle, which is uniformly distributed throughout a user-defined range. This means that the position the firefly can end up sweeps out part of a circle (aka a sector) with angle θ. Next, the firefly takes a step (step size is another user-defined parameter) and repeats this process. Within this process is embedded the flashing strategy for signaling: A firefly has a defined flash interval (the number of steps between flashes) and uses it throughout its trajectory to turn on and off its flashing.

I am satisfied with the theoretical basis for now, so I have also started to implement this model into an OOAD framework. So far, I followed my class diagram in Homework 4 while changing things to what makes more sense as I code. I have an abstract class `Firefly` with attributes for firefly sex and x and y positions. The concrete class `MaleFirefly` inherits from `Firefly` while having its own specific attributes (such as an id number, number of total steps, step size, flash interval, and turning angle distribution). The behaviors of a male firefly are also fairly simple, captured in these methods: `pick_turning_angle()`, `check_flash()`, `take_step()` (The code has more detailed comments on the functionality of each method). Further, I have a simple factory class, `FireflyFactory`, to create a `MaleFirefly` object given the parameters. Then, the `FireflyCollection` is for now used to generate many and to contain all fireflies created by the factory. `FireflyCollection` asks `FireflyFactory` to create `MaleFirefly` objects.

Right now, I simple have a main space where I define parameters and instantiate a firefly collection to generate/initialize fireflies to be in the simulation. I have been testing the model by running short simulations to check for technical and conceptual bugs. In the global space I also have code to generate a movie of a firefly sky, with the trajectories and flashing patterns of the firefly visible. There's also an option to turn off the trajectories and only have the flashes, which epitomizes the difficulty of a female picking out and tracking a single male to respond to his signal and mate. Below are some screenshots of these movies. Here's a link to a movie with trajectories and flashing, and another movie with only flashes in the sky. The GitHub repo for the code in development is here (sorry, still messy).





What's left:

Everything currently in the global space needs to be encapsulated into appropriate classes. Depending on timing, I may or may not also model the female firefly behavior (or just model a simple iteration of it for now). Mostly, I want to focus on making the user interface, and thus the `SimulationSpace`, `Observer`, `MainSimulator`, `UserInterface` classes mentioned in Homework 4. I am hoping to make an interface where the user can learn the background

behind the model, enter parameter they would like to test (perhaps using interactive widgets like sliders), and be able to see the results of the parameters they chose. I still plan to do everything in Jupyter Notebook with Python, and will be exploring widget and animation libraries.

Plans for next two weeks

Some concrete goals to achieve:

- Encapsulate the code in global space!
- Implement a simple strategy of how females track a male and predict his position.
- Implement a simulator/environment class.
- User interface will be a single Jupyter notebook which imports other notebook with model implementations. For this interface, try to use widgets so user can input parameters within a range, choose to save data, etc. Also, here, display some simulation results in form of a movie, embedded in the notebook if possible.

Class Diagram