

Report for Final Project: Bobbles and Friggles

-Jonas Bartels

Bobbles and Friggles is a simulation that simulates natural species interacting in their environment. In it, Bobbles (carnivores), and Friggles (herbivores) survive on their landscape. They live on a grid composed of land plots. Beeterweed (the food friggles survive off of) grows on these plots.

This simulation allows the user to make a variety of changes to the behavior of beeterweed, friggles, and bobbles. It also gives the user free range when it comes to the dimensions of the simulation and the amount of bobbles/friggles it starts out with. The effects of changes in these variables can then be observed by running the simulation.

Instructions for running the simulation:

Setup

To run the simulation you need only enter: **"javac *.java"** after a short compiling, the console will jump to the next line. In this next line you will be inputting parameters for the simulation. X is the width of the simulation(in land plots) and Y is the height of the simulation. N_f will be the amount of friggles at the start and N_b will be the amount of bobbles at the start. Finally, D will be either 0,1,or 2. D determines how the simulation is displayed to the user. $D = 1$ will show friggles and bobbles in a grid. $D = 2$ will show friggles and bobbles but it will hide the gridlines making it a bit easier to interpret what is happening if the simulation is looping quickly. If you are working on a larger simulation (bigger than 30×30) the display may become distorted due to a lack of space in the console causing text-wrapping. To avoid this, and the long extra wait time on huge(1000×1000) simulations, you can set D to 0 which will result in no visual depiction of the land area being given.

These variable parameters should be inputted into the simulation in this format:

"java Simulation X Y N_f N_b D"

Now you have started the simulation.

In the console you should be able to see Facts about the friggles, bobbles, and land plots (we'll get to changing these parameters later)

If you chose display option (D) 1 or 2, you should also see three fields. The leftmost field shows beeterweed, the middle field shows the friggle population, and the right of the three shows the bobble population. The numbers indicate how many of a certain type are on a certain land plot It should look similar to this

```
❯ javac *.java
❯ java Simulation 10 15 20 20 2
```

Friggle Facts:

Maximum Age:12

Starting Energy Level:5

Distance covered per step (distance between landplots):2

steps taken per day (landplots visited per day):2

Size of litter (when reproduction occurs): 3

Energy level required to reproduce: 10

Bobble Facts:

Maximum Age: 20

Starting Energy Level:30

Distance covered per step (distance between landplots): 8

steps taken per day (landplots visited per day):9

Size of litter (when reproduction occurs):1

Energy level required to reproduce:22

Land Plot Facts:

Time it takes beeterweed to grow back:2

[illegible]

Time steps to simulate:

Running simulation timesteps

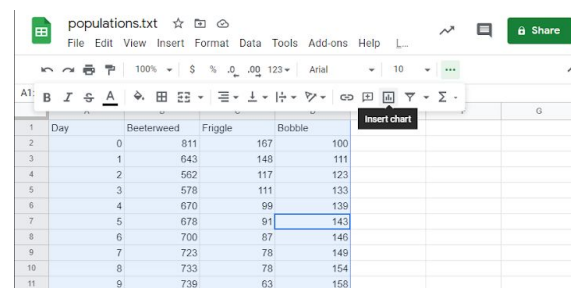
At the bottom, you will see a prompt that says “Timesteps to simulate (or save):”. Enter a positive integer number and hit return/enter. The simulation will then start looping that amount of times. This process can be repeated.

Saving population information

When you are satisfied with the simulation, enter “save” instead of a number and the population data from the simulation will be transferred to a new file called *populations0.txt*. If you run the simulation again, it will generate a *populations1.txt*. It will incrementally increase so that no data is lost. Note: a file will always be numbered with the lowest available number.

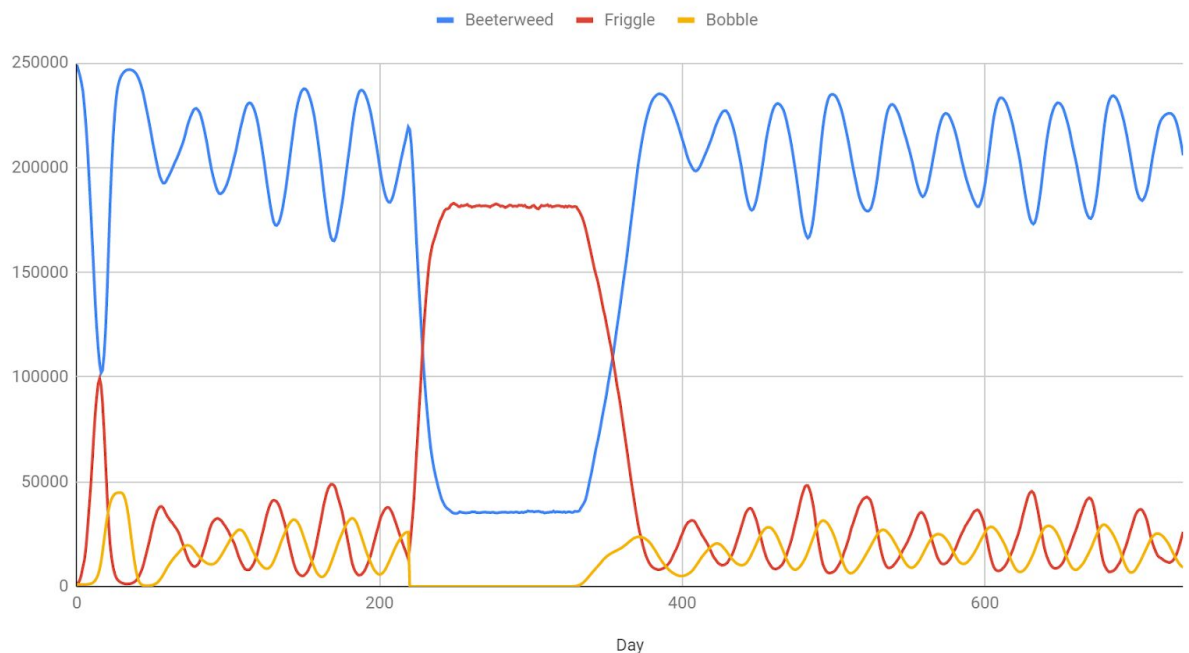
Visualize with google sheets:

Google sheets has a method of easily transferring the data from *populations0.txt* to a google sheet, which can easily be visualized as a graph. To do this, open a new google sheet, under file, select import. From there, select Upload. Drag the *populations0.txt* file to the box. Once that is done, select any field with data in it, hit Ctrl + A, and press the insert chart button.



Day	Beeterweed	Friggle	Bobble
0	811	167	100
1	643	148	111
2	562	117	123
3	578	111	133
4	670	99	139
5	678	91	143
6	700	87	146
7	723	78	149
8	733	78	154
9	739	63	158

Beeterweed, Friggle and Bobble



Yellowstone features

An extra feature that was added is the “exterminateBobbles” and “reintroduceBobbles” commands. They can be used in place of a digit or “save” for the same prompt. They were added to demonstrate the harm that removing predators from an ecosystem can have on the vegetation. (inspired by the wolves at yellowstone) this is also visible in the graph above

Changing animal attributes

It is also possible to change the attributes of bobbles, friggles, and land plots. To do this, look into the directory and open *Bobble Facts.txt*, *Friggle Facts.txt*, or *Land Plot Facts.txt*. When you change these numbers, remember that they must stay integers (no decimals) and that the number must directly follow the colon.(no spaces in front) so that the simulation can read in those numbers correctly.

Energy: energy represents health/size/strength. It increases when an animal eats. (a friggle gains 3 when it eats, when a bobble eats a friggle, it increases its energy by the amount of energy that friggle had). If an animal reaches an energy level of 0, it dies (of starvation).

Reproduction:

Reproduction requires a certain minimum energy level, individual to each species. When an animal reproduces it splits its energy level among itself and its offspring evenly.

Movement: when an animal moves, it visits a set number of land plots and tries to graze/hunt at each one. There is one parameter for how many land plots are visited and one parameter for how far apart two consecutively visited land plots may be.

.

What are the superclass and two subclasses your project uses?

In my code I use a superclass called Animal. My two subclasses are Bobble and Friggle

Why is inheritance useful for your superclass and subclasses?

Friggles and Bobbles share many different variables and methods (like moving,energy management, and a load of setters and getters) To avoid writing unnecessary code and making

debugging easier, inheritance was the way to go. It would also vastly make adding new animals, or the possibility of attribute mutation in the future, much easier. It would be a method in `Animal.java` that would take in a variable and a double. Then use that double as a chance for whether that variable changes and by how much. Then it would return the new value which would then be used when an animal is “materialized.”

What is the other class concept your project uses and why is that concept the best to use in your project?

I also used a priority queue. When a Bobble hunts, it checks the land plot that it's on to see if there are any frigglers there. If a land plot could only house one friggler, the priority queue would be useless. This is not the case. A land plot could have many frigglers on it. So when a bobble comes to hunt on a land plot, it will eat the weakest(lowest energy level) friggler in that location because that is the friggler that would be easiest to hunt. I insure this by having each land plot have a Friggler priority queue with the weakest friggler at the top. I could also just sort the frigglers at one land plot by weakness but that would be far less time efficient as a priority queue only requires a couple adjustments when objects are added/taken. This becomes important with really large simulations because the time this simulation takes is a function of the amount of land plots and animals. The use of the priority queue also will add more interesting dynamics if I choose to add mutation later on. It may drive frigglers to increase their reproduction energy levels so that they will have a better chance of survival.