448.057 Context-Aware Computing (UE)

Exercise 2 – Farmer breeding Goats

Task description

The second system to be modelled using NetLogo consists of a farmer breeding goats. The goal is to automatically adapt the parameters of the model such that the farmer minimizes the resources (and hence the costs) necessary to feed the goats, while ensuring their survival.

The farmer owns a piece of land that should be modelled as a 45x45 torus field (i.e., the virtual world wraps horizontally and vertically). The field is a brown piece of land with some randomly dispersed green grass patches. The amount of grass on the remaining land is ruled by the input variable `initial-grass-density`, whose range is $[1-99]\%$.

The farmer initially owns `goats-number` goats, where `goats-number` is a variable specified by the user in the range $[1-100]$. The goats are initially randomly dispersed in the field (either in the grass or on the brown patches) and have a "sheep" shape, white color, and a size of 2.

Goats eat the grass from the field to survive, so they move around the field in a random direction looking for some grass to eat. As soon as they find a patch with some grass, the goats eat the grass, leaving the patch empty (brown), i.e., the amount of grass in the farm will decrease as soon as the goats start moving around and eating it.

Every goat has an amount of energy available for movements. Each time a goat moves one patch away, it consumes `energy_expenditure` energy, with `energy_expenditure` being a variable specified by the user in the range $[1-50]$. When a goat eats grass, it acquires some energy. One patch of grass corresponds to `acquired_energy` energy. Also the `acquired_energy` variable is specified by the user and its range is $[1-50]$. When the simulation begins, the goats should have an initial amount of energy equal to `initial_goats_energy`, with the latter being an input variable taken through a slider with range $[1-50]$. The energy is spent only by moving around: a goat that does not have any more energy (zero or below) dies.

Use the following code in your movement function:

Listing 1: Example for movement function

```
rt (20 - random 40); Change the heading by +/- 20 degree
fd 1 ; Move one step forward
```

Listing 2: Skeleton for the setup procedure.

```
to setup
        clear-all
        reset-ticks
        setup-brown-land
        setup-grass
        setup-goats
end
```


Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

The initial grass dispersed in the farm is obviously not sufficient for all the goats to live forever. For this reason, the farmer has to refill the field with some grass periodically to avoid that some goats die. This is modelled in the procedure `generate-grass` that produces new green patches at random positions. The number of generated grass patches is initially controlled using a `new_grass` slider with range $[0 - 50]$.

The simulation should contain two buttons: `setup` and `go`, associated to the `setup` and `go` procedures, respectively. The `setup` procedure creates the brown field and generates an initial amount of grass and goats. These four operations are executed inside the procedures `setup-brown-land`, `setup-grass`, and `setup-goats`, respectively, as shown in Listing 2.

The `go` procedure models the daily activities ongoing in the farm, where one day corresponds to one simulation round (one tick). Every morning the farmer spreads some grass out in the field, and immediately afterwards the goats will start looking for grass and eating it. Within one simulation round, the goats will therefore continue moving until they will either find a piece of grass to eat or they will starve because they are "exhausted" (i.e., they have no more energy). Please notice that goats cannot eat more than one piece of grass per day, i.e., once they have eaten a piece of grass, they rest in the same position until the next day.

At the end of each day, the farmer checks the status of the field, and updates its diary with some statistics. This is done by updating four plots and three monitors. The four plots show graphically (i) the amount of goats left, (ii) the amount of grass patches left, (iii) the average amount of energy of each living goat, and (iv) the average number of steps it takes the goats to find grass. The three monitors show the amount of goats, as well as the number of brown and green patches in the field.

Basic functionality *(4 points)*. Create a NetLogo model that follows the specifications above and make sure that the program does not enter an infinite loop. Moreover, it should contain a *chooser* to select between the different methods to calculate the new amount of grass. The manual method should be called `manual`, the other two `adaptive1` and `adaptive2`. The submitted code should be properly commented and each team member should be able to explain it.

System behaviour and modelling *(3 points)*. Discuss the high-level behaviour of the system by answering the following questions (please respect the space constraints):

1. Propose a set of input values for which all goats survive without getting too fat (i.e., their energy converges to a reasonably constant value).

   Answer:

   The best possible scenario we were able to find was fot this set:

   Number of goats: 19
   Initial grass density: 56%
   New grass: 19
   Initial goats energy: 35
   Acquired energy: 2
   Energy expenditure: 1

2. Model the expected number of steps the goats travel to find food as a function of green and brown patches. Give your formula.

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

Answer:

We used the formula for the expected number of failures (in our case steps) before the first success for the geometric distribution.

E(Y) = (1 - p)/p,

where p is the probability of hitting a green patch in each step

p = the number of green patches / the total number of patches

3. Model how the mean energy of the goats changes per simulation round as a function of the `energy_expenditure`, `acquired_energy`, and the expected number of steps from the previous question. (Hint: Read page 3 and 4 of [1]). Give your formula.

Answer:

Our formula:

E(X) = E(X) - E(Y) * energy_expenditure + acquired_energy,

where E(X) is initialized to value of initial_goats_energy and E(Y) is the expected number of steps.

Adaptation ***(3 points)*****.** Adaptively minimize the costs for the farmer as follows:

1. Model the relationship between `new_grass`, `energy_expenditure`, `acquired_energy`, the current number of goats, and the average steps: find a formula to calculate how much grass the farmer should "plant" every round, in such a way that the goats will survive without getting too fat for around 10.000 rounds. Consider using the formulas from the previous questions. The formula should be used in a procedure called `adjust-grass` that extracts the system state by reading the current number of goats, as well as the `energy_expenditure`, `acquired_energy` variables and the number of brown and green patches. It then computes the optimal value for `new_grass` using your formula in every round. Activate this method when the option `adaptive1` is selected in the chooser.

Answer:

2. Discuss if it is always possible to let the goats converge to a reasonably constant energy with your approach. If yes, how? If not, why not?

[1] `http://theory.bio.uu.nl/rdb/books/mpd.pdf`]

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

Answer:

3. Extend your algorithm such that the energy level of the goats is taken into account to calculate the number of new grass. What are the advantages over your previous solution? Activate this method when the `adaptive2` option is selected in the chooser.

   Answer:

Please submit the filled PDF and your NetLogo source code to the TeachCenter as illustrated in the lecture within **Friday, 23.11.2018, 23:59 CET**. Make sure to use a switch or chooser to enable or disable the "adaptation" functionality.

Group ID:	13
Stud. Names:	Juraj Vida Timur Cerimagic

Finalize Form

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>