Exercise 1: Implementing a first Application in RePast: A Rabbits Grass Simulation

Group 43: Michael Martini, Philine Witzig

28th September 2020

PROBLEM REPRESENTATION

In this exercise we were asked to implement a first application in RePast, an open-source Java Toolkit for programming simulations. In particular, the goal was to simulate an eco-system of rabbits and grass. More precisely, this eco-system was represented through a discretized grid-world (with default size of 20×20), where grass is randomly growing (with variable rate) on the different grid cells. Rabbits move around this world randomly (possible moves: NORTH, SOUTH, EAST, WEST) where each move comes with a loss of energy. If a rabbit moves to a cell which contains grass, it can eat the grass which leads to an increase of the rabbit's energy. Once a certain energy threshold is reached, a rabbit will reproduce, which in turn causes a reduction in energy. Overall, this lead to the following parameters for which we tested different combinations in the ranges mentioned below:

- *BirthThreshold* $\in [10, 100]$
- *GrassGrowthRate* $\in [0, 1000]$
- *GridSize* $\in [10, 100]$ (grid is always quadratic)
- *NumInitGrass* $\in [0, 4000]$
- *NumInitRabbits* $\in [10, 150]$

Note that we filter for invalid parameter combinations, e.g. *NumInitRabbits* $>$ *GridSize* $\times$ *GridSize* and throw exceptions for the respective combinations. Moreover, we set a cap to the grass size to ensure that grass cannot grow infinitely large.

IMPLEMENTATION DETAILS

We followed the given tutorial closely adapting it for the given simulation task at the required places. Our main changes included the different implementation of movement of the agents and the implementation of a birth logic in comparison to the constant population state of the Carry-Drop Simulation. The movement control uses a switch-case statement to choose the next cell to move to (with wrap around on the edges) in a uniform fashion using a random number. New agents are born if their energy is higher than the required energy threshold (cf. *BirthThreshold*). The original agent looses energy (i.e. current_energy - *BirthThreshold*) while a new agent is created starting with *StartingEnergy* = 15.

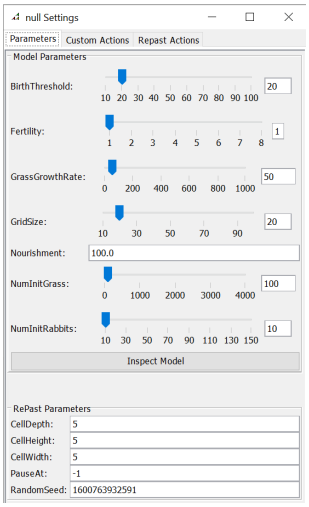
We added two additional parameters, one representing the nourishment of the grass that is eaten, i.e the percentile of absorbed energy by the rabbit, as a mutable value. Furthermore, we added a fertility parameter representing the amounts of rabbits born as the uniform selection of a number of rabbits born. We deemed this as interesting, as rabbits fall in the category of r-strategist so adding this to the simulation allows us to observe the impact.

Furthermore, we added the requested sliders, both, to the initialization and as custom actions in the actions tab. Besides the sliders as requiered for the parameters *GridSize*, *NumInitRabbits*, *NumInitGrass* and *GrassGrowthRate*, we also added sliders for values that we expanded on or introduced in our implementation such as *Nourishment* and *Fertility*, having the values below:

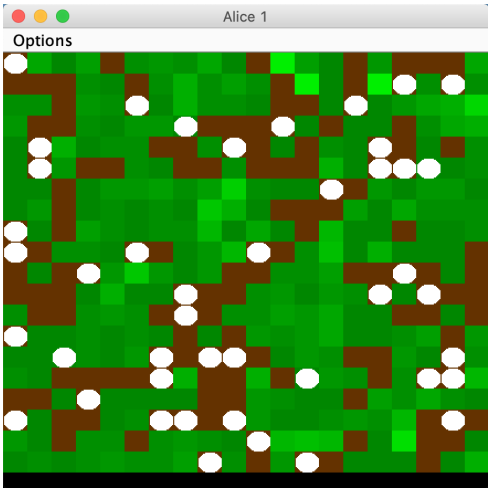
- *Fertility* $\in [1, 8]$
- *Nourishment* $\in [0, 200]$.

Confer Figure 1a for a visualization of the RePast UI with the respective sliders. Furthermore, Figure 1b shows the visualization of the simulation space.

For the requested plot, we followed the tutorial and implemented it based on RePast, plotting both the overall grass and rabbit population of the simulation space for every 10'th simulation step. A sample plot



(A) Screenshot of the RePast UI with sliders for the model parameters.



(B) Screenshot of RePast UI with visualization of the simulation space.

FIGURE 1
Screenshots of the RePast UI.

can be seen in Figure 2.

RESULTS

We experimented with different parameter settings to observe the relation between the model parameters. What can already be seen in Figure 2 is that the rabbit population can change based on the overall availability of grass, as an increase in grass leads to a slightly delayed increase in rabbits. However, this is only the case if the *StartingEnergy* is below the *BirthThreshold*. Otherwise, the rabbits can survive without grass (cf. Figure 3a) as the number of rabbits converges to a value unequal to 0 while there is no grass left in the simulation space. Another interesting aspect is that the population dies although there is still grass available if the grid is too large and the individuals loose all of their energy before they reach a grass cell (cf. Figure 3b).

Additionally, we investigated the effect of the parameters which we added, i.e. *Nourishment* and *Fertility*. If *Nourishment* is set to its minimum while all other parameters are set to their default values, the population will also eventually die out while the grass keeps growing until all cells are overgrown with grass. Setting it to its maximum value does not have too much of an effect since the grass availability due to the default values keeps the population in place. The same can be observed when changing the *Fertility* parameter to its maximum. Again, the default values of the other parameters ensure that the grass availability keeps the rabbit population constant. When setting the *Fertility* parameter to its minimum value, we obtain the graph in Figure 2.

To conclude, one can say that we tried to simulate a real rabbit-grass-world as well as possible by adding constraints where necessary, e.g. grass cannot grow infinitely large, there cannot be more grass or rabbits than there is space etc. On the other hand, we ensure that the user has multiple reference points for controlling the circumstances of the rabbit-grass-world to discover different population behavior. One minor improvement could be to avoid cases like in the run represented in Figure 3a, where rabbits survive without grass. However, we kept this case to show the limitations of such a simulation when not restricting parameters properly.

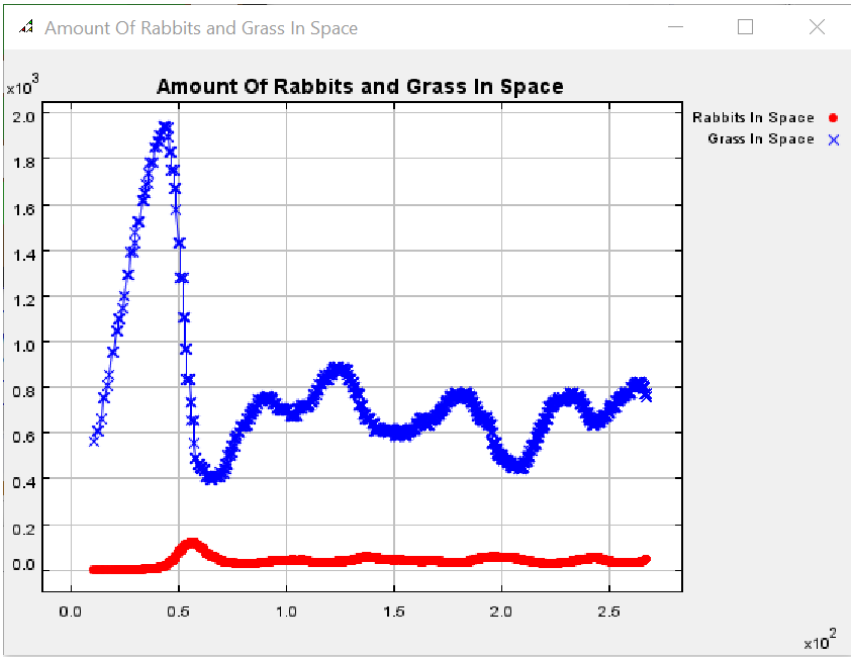
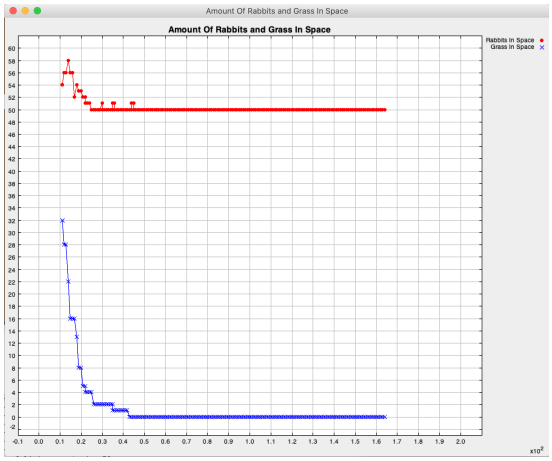


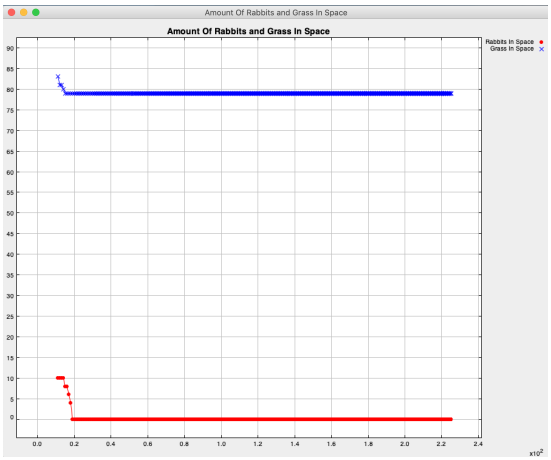
FIGURE 2
Sample plot showing the development of the numbers of rabbits and grass in the simulation space over time. The x-axis represents the the number of simulation steps (in 10^2), while the y-axis captures the number of rabbits (blue) and grass (red) respectively (in 10^3)

.



(A) RePast plot with *BirthThreshold* = 10 < *StartingEnergy* = 15, *GrassGrowthRate* = 0, *Fertility* = 8, all other parameters set to default values

.



(B) RePast plot with *BirthThreshold* = 100 > *StartingEnergy* = 15, *GrassGrowthRate* = 0, *Fertility* = 8, all other parameters set to default values

.