THE ENVIRONMENTAL PROTECTION AGENCY OF PYLANDIA

Modelling the Ecosystem of Rossumøya

Dr. Hans Ekkehard Plesser
DIRECTOR, ENVIRONMENTAL PROTECTION AGENCY OF PYLANDIA

Jørgen Kvalberg
SPECIAL ADVISER

Version 1.2: 2016-01-15

Project in Advanced Programming at IMT / NMBU, January 2016.

1 Introduction

Rossumøya is a small island in the middle of the vast ocean that belongs to the island nation of Pylandia. The ecosystem on Rossumøya is relatively undisturbed, but Pylandia's Environmental Protection Agency wants to study the stability of the ecosystem. The long term goal is to preserve Rossumøya as a nature park for future generations.

The ecosystem on Rossumøya is characterized by several different landscape types, jungle, savannah, mountains and desert. The fauna includes only two species, one species of herbivores (plant eaters), and one of carnivores (predators). You shall investigate whether both species can survive in the long term. A detailed description of Rossumøya's geography and fauna is given in section 2. The most important characteristics are

Herbivores depend on a good supply of fodder to survive and reproduce.

Carnivores depend on the availability of prey. Carnivores are more mobile than herbivores.

Jungle provides large amounts of fodder even under intense grazing.

Savannah can be destroyed by overgrazing.

Desert does not provide fodder for herbivores.

Mountain are impassable for herbi- and carnivores.

Ocean is impassable for both species.

A map of Rossumøya is shown in Figure 1.

1.1 Project task

The Environmental Protection Agency of Pylandia (EPAP) encourages research groups to develop computer programs for the simulation of population dynamics on Rossumøya. EPAP expects that

- **groups** of **two** experts
- develop a **population dynamics simulation**
- by **Wednesday, 20 January 2015, 16.00**.

EPAP will hold regular information seminars on the project during the project period. EPAP expects interim reports in accordance with the milestones shown in Table 1.

Expert groups will by 20 January 2016 deliver source code and documentation to EPAP. Details on how code and documentation are to be delivered will be provided later.

Since it is encountered political controversy about the reliability of the simulations, EPAP places great emphasis on the quality of the delivery. All code development must be traceable through regular commits to a version control system, including all exchange of code between team members. Code should only be written once suitable unit tests are in place and committed.

All participants must individually present their code and illustrative results during interviews held on 25 January 2016. The details will be presented at a later date.

2 The Nature of Rossumøya

2.1 Geography

Rossumøya is divided into squares (or cells), and each square is of one of the following five types:

Date	Milestone
04 Jan 2016	Startup: Setup of Bitbucket Repository/Branch and PyCharm Project
05 Jan 2016	Problem analysis and requirements analysis.
08 Jan 2016	Demonstration of a first functional simulation of *herbivores* in one place (no migration).
11 Jan 2016	Project plan for remaining work.
15 Jan 2016	Demonstration of a working simulation of herbivores and carnivores, all types of terrain and simple visualization.
20 Jan 2016	**16.00:** Full simulation code with documentation.
24 Jan 2016	**12.00:** PDF and animation for oral presentation.
25 Jan 2016	**08.00–19.00:** Individual presentations and assessment interviews.

Table 1: Project milestones. All milestones are at 16.00 unless other times are given explicitly.

- ocean,
- mountain,
- desert,
- savannah,
- jungle.

As an island, Rossumøya is surrounded by sea: cells on the outer edges of the map of Rossumøya (see Fig. 1) are therefore always of type "Ocean". Animals can only move from the square they are in to one of

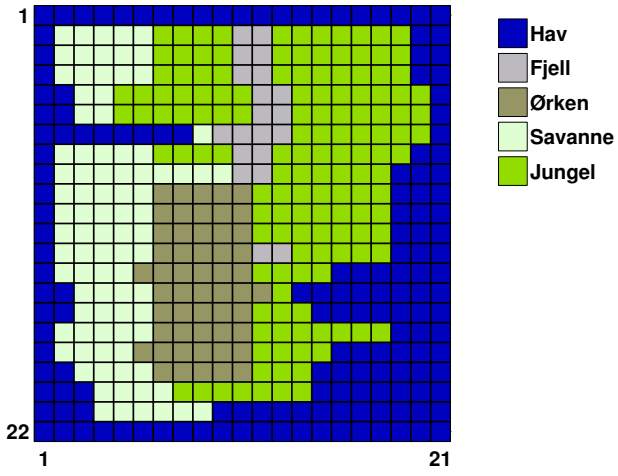

Figure 1: Landscape types on Rossumøya according to the last survey.

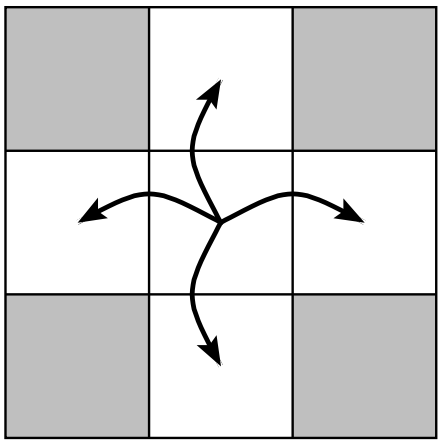

Figure 2: An animal that is in the middle cell can move to one of the four neighboring cells but not to cells diagonally displaced (gray).

the four nearest neighbor squares, see Fig. 2. No animal may stay in the ocean or the mountains.

2.1.1 Ocean and mountains

Neither the ocean or mountains can be entered by animals. Cells of these two types will be completely passive in the simulation.

2.1.2 Desert

Animals may stay in the desert, but there is no fodder available to herbivores there. Carnivores can prey on herbivores in the desert.

2.1.3 Savannah

The savannah offers fodder to herbivores in limited quantity and is sensitive to overgrazing. In each cell there is a certain amount f_{ij} of fodder, where the indices (i, j) determine the cell. Every time a herbivore eats, the animal tries to consume a certain amount F of fodder. The following "eating rules" apply:

if $F \leq f_{ij}$
  there is enough fodder, the animal eats F and
  the amount f_{ij} of fodder in the cell is reduced by F
if $0 < f_{ij} < F$
  the animal eats is what is left of fodder, i.e., f_{ij},
  and f_{ij} is set to 0
if $f_{ij} = 0$
  the animal receives no food.

Every year new fodder grows according to (see also Sec. 2.3)

$$f_{ij} \leftarrow f_{ij} + \alpha \times (f_{\max}^{\text{Sav}} - f_{ij})\,. \tag{1}$$

Carnivores can prey on herbivores in the savannah.

2.1.4 Jungle

In the jungle, mostly the same "eating rules" apply as in the savannah, see section 2.1.3. The growth of vegetation in the jungle is, however, so quick that the jungle is not susceptible to long term damage due to overgrazing. Each year, a fixed amount of fodder is available (see also Sec. 2.3):

$$f_{ij} \leftarrow f_{\max}^{\text{Jungle}} . \tag{2}$$

Carnivores can prey on herbivores in the jungle.

2.2 Fauna

Herbivores and carnivores have certain characteristics in common, but feed in different ways, see section 2.2.1 and 2.2.2. The similarities are as follows[1]:

1. Age At birth, each animal has age $a = 0$. Age increases by one year for each year that passes.

2. Weight Animals are born with weight $w \sim \mathcal{N}(w_{\text{birth}}, \sigma_{\text{birth}})$, i.e., the birth weight is drawn from a Gaussian distribution with mean w_{birth} and standard deviation σ_{birth}. When an animal eats an amount F of fodder, its weight increases by βF. Every year, the weight of the animal decreases by ηw.

3. Fitness The overall condition of the animal is described by its fitness, which is calculated based on age and weight using the following formula

$$\Phi = q^{+}(a, a_{\frac{1}{2}}, \phi_{\text{age}}) \times q^{-}(w, w_{\frac{1}{2}}, \phi_{\text{weight}}) \tag{3}$$

where

$$q^{\pm}(x, x_{\frac{1}{2}}, \phi) = \frac{1}{1 + e^{\pm\phi(x - x_{\frac{1}{2}})}} . \tag{4}$$

Note that $0 \leq \Phi \leq 1$.

4. Migration Animals migrate depending on their own fitness and the availability of fodder in neighboring cells. Animals can only move to the four immediately adjacent cells. Animals cannot move to ocean or mountain cells.

An animal moves with probability $\mu\Phi$.

If an animal moves, the destination cell is chosen depending on the amount of fodder available in neighboring cells as follows. Let i be the cell the animal is in now, and let $\mathcal{C}^{(i)}$ be the set of its four next-neighbor cells[2]. For each cell, we define the relative abundance of fodder

$$\epsilon_k = \frac{f_k}{(n_k + 1)F} , \tag{5}$$

where f_k is the amount of relevant fodder available in cell k, n_k the number of animals of the same species in cell k and F the "appetite" of the species. "Relevant fodder" is the amount of plant fodder available if the moving aninmal is a herbivore, and the total weight of all herbivores in cell k if the moving animal is a carnivore.

Then, the *propensity* to move from i to $j \in \mathcal{C}^{(i)}$ is given by

$$\pi_{i\to j} = \begin{cases} 0 & \text{if } j \text{ is Mountain or Ocean} \\ e^{\lambda\epsilon_j} & \text{otherwise} \end{cases} \tag{6}$$

and the corresponding *probability* to move from i to j is given by

$$p_{i\to j} = \frac{\pi_{i\to j}}{\sum_{j\in\mathcal{C}^{(i)}} \pi_{i\to j}} . \tag{7}$$

In the special case where all next neighbors of a cell are Mountain or Ocean cells, so that $\pi_{i\to j} = 0$ for all $j \in \mathcal{C}^{(i)}$, the animal does not walk.

For $\lambda = 0$, all possible destination cells will be chosen with equal probability. For $\lambda > 0$, the animal will prefer cells with the greater abundance of food, while for $\lambda < 0$ it would turn away from food.

An animal can walk only once per year.

5. Birth Animals can mate if there are at least two animals of the same species in a cell.

- For each animal in a cell, the probability to give birth to an offspring in a year is

$$\gamma \times \Phi \times (N - 1) , \tag{8}$$

  where N is the number of animals of the same species in the cell at the start of the breeding season.
- The probability is therefore 0 if there is only one individual of the species in the cell.
- The probability of birth is also 0 if the weight is $w < \zeta(w_{\text{birth}} + \sigma_{\text{birth}})$.
- If Eq. (8) gives a probability greater than 1, set the probability to 1, ie the parent animal gives birth with certainty.
- It is assumed that gender does not play a role in mating.
- Each animal can give birth to at most one child per year.
- At birth, the mother animal loses ξ times the actual birthweight of the baby.

6. Death Animals may die. An animal dies with probability

$$\omega(1 - \Phi) \tag{9}$$

where $0 \leq \omega \leq 1$

[1] See section 2.3 for details on how the various processes are distributed throughout the year.

[2] To keep notation simple, we use linear indices to the cells here instead of coordinates in two dimensions. Simply consider enumerating all cells sequentially starting with the top-left cell.

Parameter	Herbivores	Carnivores
w_{birth}	8.0	6.0
σ_{birth}	1.5	1.0
β	0.9	0.75
η	0.05	0.125
$a_{\frac{1}{2}}$	40.0	60.0
ϕ_{age}	0.2	0.4
$w_{\frac{1}{2}}$	10.0	4.0
ϕ_{weight}	0.1	0.4
μ	0.25	0.4
λ	1.0	1.0
γ	0.2	0.8
ζ	3.5	3.5
ξ	1.2	1.1
ω	0.4	0.9
F	10.0	50.0
$\Delta\Phi_{\text{max}}$	—	10.0
	Jungle	Savanne
f_{max}	800.0	300.0
α	—	0.3

Table 2: Example values for parameters of herbivores and carnivores, as well as for Jungle and Savannah. These values are used in the examples shown in the text, but have no special meaning.

The parameters w_{birth}, σ_{birth}, β, η, $a_{\frac{1}{2}}$, $w_{\frac{1}{2}}$, ϕ_{age}, ϕ_{weight}, μ, λ, γ, ζ, ξ, ω, F, and $\Delta\Phi_{\text{max}}$ are identical for all animals of the same species, but may be different between herbivores and carnivores. Example values are given in Table 2.

2.2.1 Herbivores

Herbivores find fodder exclusively in the savannah and jungle. Animals residing in a cell eat in the order of their fitness Φ, with the animal with the highest fitness eating first. The animal tries every year to eat an amount F of fodder, but how much feed the animal obtain depends on fodder available in the cell, see section 2.1. Given that the animal eats an amount $\tilde{F}$ of fodder, its weight increases by $\beta\tilde{F}$.

2.2.2 Carnivores

Carnivores can prey on herbivores everywhere, but do not prey on each other. Carnivores try to kill herbivores in the order of fitness, i.e., the carnivore with the highest fitness eats first. Carnivores try to kill one herbivore at a time, beginning with the herbivore with the lowest fitness. A carnivore continues to kill herbivores until

- the carnivore has eaten an amount F, i.e., eaten carnivores with a total weight $\geq F$
- or has tried to kill each herbivore in the cell.

Carnivores will kill a herbivore with probability

$$p = \begin{cases} 0 & \text{if } \Phi_{\text{carn}} \leq \Phi_{\text{herb}} \\ \frac{\Phi_{\text{carn}} - \Phi_{\text{herb}}}{\Delta\Phi_{\text{max}}} & \text{if } 0 < \Phi_{\text{carn}} - \Phi_{\text{herb}} < \Delta\Phi_{\text{max}} \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

The carnivore's weight increases by βw_{herb}, where w_{herb} is the weight of the herbivore killed[3]. The fitness of the carnivore is re-evaluated each time it kills a herbivore.

2.3 The Annual Cycle on Rossumøya

Nature on Rossumøya follows a fixed annual cycle. The components of annual cycle are:

1. **Feeding** Animals eat: first herbivores, then carnivores, see section 2.2.1 and 2.2.2. Growth of fodder in savannah and jungle according to Eqs. (1)–(2) occurs at the very beginning of the year, i.e., immediately before herbivores eat.

2. **Procreation** Animals give birth, see section 2.2, No. 5. When calculating the probability of birth according to equation (8), the number of animals N at the start of the breeding season is used, i.e., newborn animals do not count.

3. **Migration** Animals migrate to neighboring cells subject to the following conditions:

    1. Migration happens at random across the island. In practice, this means that migration out of each cell is simulated in a different random order of cells for each year.
    2. For each cell, migration is first simulated for all herbivores, then for all carnivores.
    3. When calculating relative abundance of food and probability of movement according to equations (5) and (7), the number of animals n_k and available fodder f_k in potential destination cell k are the numbers at the time a given animal moves, i.e., they include all animals that have moved to cell k at earlier times in the current year.
    4. Each animal can walk only once per year.

4. **Aging** Each animal becomes one year older.

5. **Loss of weight** All animals lose weight, see section 2.2, No. 2.

6. **Death** For each animal, it is determined whether the animal dies or not, see section 2.2, No. 6.

Steps 1–6 can be seen as the seasons on Rossumøya, i.e., all animals undergo steps simultaneously.

[3] If the weight of the herbivore exceeds the amount of food desired by the carnivore, the carnivore eats only the amount it wants. The remainder of the herbivore goes to waste.

2.4 Fodder available during the First Year

At the start of the simulation, all jungle cells contain an amount $f_{\max}^{\text{Jungle}}$ of fodder, while all savannah cells will contain an amount $f_{\max}^{\text{Sav}}$.

3 Guidelines

3.1 Development process

All source code shall be managed on a Git repository hosted on Bitbucket. EPAP representatives shall have read access to the repository at all times during the development process.

Code changes shall be committed to the repository in small increments, and all code exchange between team members shall be via the team repository.

Basic agile programming principles shall be followed, especially a focus on pair programming and test-driven development using nosetests.

3.2 Deliverables

Each team will deliver the following:

Software Simulation Software in the form of a Python package compatible with Python 2.7. The package should be organized so that a normal user can carry out simulations using the methods of a class, which forms the interface for the package. The package should be installable using the Python distribution tools. The program code shall be well-structured, documented, and efficient. The package must contain unit and integration tests covering the code. Coding shall follow PEP8 guidelines. **The compatibility check script given in Appendix B shall work with your package.**

Documentation The software shall be well-documented. All packages, modules, classes and public methods must be documented with docstrings. In addition, browsable user documentation shall be created with the Sphinx tool, allowing domain experts to use the software.

Presentation Software and exemplary results will be presented in an individual oral presentation of 5 minutes duration (one PDF file and one animation file). The presentation shall discuss the main aspects of the chosen implementation, its advantages and disadvantages.

3.3 Parameters and Initialization

3.3.1 Geography

The software should be able to read a Python multiline string specification of the island's geography. All lines in the string must have the same length. Each character in the string represents a cell with character code shown in Table 3.

O	ocean
J	jungle
S	savannah
D	desert
M	mountain

Table 3: Codes of landscape types.

Parameter	names
w_{birth}	`w_birth`
σ_{birth}	`sigma_birth`
β	`beta`
η	`eta`
$a_{\frac{1}{2}}$	`a_half`
ϕ_{age}	`phi_age`
$w_{\frac{1}{2}}$	`w_half`
ϕ_{weight}	`phi_weight`
μ	`mu`
λ	`lambda`
γ	`gamma`
ζ	`zeta`
ξ	`xi`
ω	`omega`
F	`F`
$\Delta\Phi_{\max}$	`DeltaPhiMax`
$f_{\max}$	`fmax`
α	`alpha`

Table 4: Names of the parameters. See Table 2 for example values.

The geography string must consist solely of "O" around the edges and no characters other than the letters shown in Table 3 must occur in the string. The software will check this.

The coordinate of the island is as illustrated in Figure 1:

- The upper left corner has coordinates $(1, 1)$.
- Coordinates increase the downward and to the right.
- The first coordinate counts the rows, the other columns.

Coordinates correspond thus to the rules for matrix elements in mathematics.

3.3.2 Parameters

It shall be possible to specify the parameters of the animal species and landscape types, respectively, by providing a dictionary of proper contents to a suitable method in the package. All in all, four dictionaries are required to provide a complete parameterization; furthermore, one method setting the parameters is required for each class (herbivores, carnivores, jungle

and savanna). Additionally, the following guidelines apply to parameters:

1. The parameter names given in Table 4 shall be used.
2. The software should have built-in default values for all parameters, so that simulations can be carried out without setting parameters.
3. It should be possible to change a subset of parameters by providing a dictionary with only those parameters that are to be changed to the method carrying out the parameterization.
4. The parameterisation methods shall report an error if a dictionary contains unknown parameters.
5. The parameterization method shall guard against wrong parameter values, such as negative amounts of fodder.

3.3.3 Populations

It shall be possible to place animals on the island before the simulation starts and during breaks in the simulation. Placement will take place through a list[4] given a suitable method in the interface class. The list shall have the following format:

1. Each item in the list is a *dictionary* with two elements, `'loc'` (Location) and `'pop'` (Population).
2. `'loc'` is a tuple with two elements and provides a coordinate on the island, see section 3.3.1. It is an error to specify nonexistent coordinates.
3. `'pop'` is a list with one element per animal.
4. Each item in `'pop'` is a dictionary with elements `'species'`, `'age'` and `'weight'`.
5. The `'species'` element has either the value `'Herbivore'` or `'Carnivore'`.
6. `'age'` shall be a non-negative integer.
7. `'weight'` shall be a positive number.

The list could, for example, look like this:

```
[{'loc': (3,4),
  'pop': [{'species': 'Herbivore',
           'age': 10, 'weight': 12.5},
          {'species': 'Herbivore',
           'age':  9, 'weight': 10.3},
          {'species': 'Carnivore',
           'age':  5, 'weight':  8.1}]},
 {'loc': (4,4),
  'pop': [{'species': 'Herbivore',
           'age': 10, 'weight': 12.5},
          {'species': 'Carnivore',
           'age':  3, 'weight':  7.3},
          {'species': 'Carnivore',
           'age':  5, 'weight':  8.1}]}]
```

[4]More precisely, an *iterable* data container.

For each list item, animals given in `'pop'` will be placed in the cell specified by `'loc'`.

The program shall ensure that

- all animals have positive weight and non-negative age, and
- animals are only place in cells where animals can stay;

If a placement violates these conditions, the program shall raise an error and terminate.

If there are animals in a cell before, then these will remain in the cell.

3.3.4 Random Numbers

Seed values for the random number generator must be set before the simulation starts. The same random number generator will be used everywhere random numbers are used in the simulation.

3.4 Simulation and Recording

3.4.1 Simulation

The simulation will run for a given number of years. After that, it shall be possible to investigate the island's status, change parameters, set out more animals, and resume the simulation for further years. One should, e.g., be able to simulate the first 100 years with only herbivores on the island, before a small group of carnivores is added and the simulation continued. The simulation will always end if there are no animals left on the island.

3.4.2 Status Information

When the simulation is stopped, it should be possible to obtain the following information in a simple way:

Year Number of years that have been simulated.

Total number of animals The total number of animals on the island.

Total number of animals by species The total number of herbivores and carnivores, respectively, in a dictionary with keys `'herbivores'` and `'carnivores'`

Per-cell animal count For each cell, the number of animals shall be available. The figures should be returned as a dictionary with keys `'herbivores'` and `' carnivores'`, where the element values are NumPy arrays with the same size as the island, where each array element is the number of animals of the given species in the respective cell.

Each type of data shall be accessible through a method in the simulation class.

3.5 Visualization

The user shall be able to visualize the simulation results while the simulation is in progress. The visualization shall be in one graphics window with the following sub-windows (*subplots*):

Geography The island's geography is shown, with a color code for the landscape types.

Total number of animals by species is shown as line graph, with one line for each species.

Population map For each species a map shall be provided showing the number of animals per cell using a color code.

Year The year shall appear in the graphics window.

Before the simulation starts, the user could specify the following:

- After how many years the graphics are to be updated (the default is each year).
- The limits for ordinate and abscissa in line graph of animal numbers.
- The limits for the color code in the population maps, separate for both species.

Furthermore, the user may specify that the graphic is saved as file after certain intervals (which must be multiples of the interval for updating the graphics). The user must also specify the start of the file name of the graphic files and ending at the filename, which also determines the file type. The files are stored must be numbered.

Note: To allow the user to convert a series of graphics files into a movie using the encoding program `ffmpeg`, files must be numbered consecutively, eg. `bs_0000.png`, `bs_0001.png`, `bs_0002.png`, ...

More information about creating movies will be provided later.

4 Further development

This project opens up many opportunities for further development. Here are some suggestions:

Optimization Get the simulation to run as quickly as possible by analyzing the time spent in the program and eliminating bottlenecks. Suggestions for optimization will be given in the January block.

Generating populations You could write auxiliary modules that generate populations based on probability distributions for weight, age and location.

Storage to file Make it possible to store data as described in section 3.4.2 to file while simulation is in progress. It should be possible to select how often data is written to the file, especially when it comes to complete information about populations.

Other information It might be interesting to compare, for example, the weight distribution of animals in the savannah with animals in the jungle. Add code to allow the necessary information to be extracted.

Serialization This means that all information about a simulation is stored in a file so that the simulation can be continued at precisely the point where it was stopped. This might be useful if you want to examine how changes affect an ecosystem: One can run the simulation with different parameter from the same starting point.

A Changes

A.1 Version 1.1: 2016-01-08

- Clarified rules for regrowth of fodder in Secs. 2.1.3, 2.1.4, and 2.3.

A.2 Version 1.2: 2016-01-15

- Removed misleading discussion of animal movement from Sec. 2.1.1.
- Clarified normalization of movement probabilites in cases where neighboring cells are Mountain or Ocean, Sec. 2.2, Eqs. (5)–(7). *This is only a formal clarification: The content of the definition is the same based provided the plausible interpretation that probabilites $p_{i\to j}$ are normalized.*
- Clarified error criteria when placing populations in Sec. 3.3.3.

B Compatibility check

The `biosim` package developed shall be compatible with the following script, i.e., the script shall execute with the package you develop. Running the script shall not require any user input. No changes shall be made to this script. The script is also available from the Course repository and shall be included in your package.

```
# -*- coding: utf-8 -*-

import textwrap
import matplotlib.pyplot as plt
from biosim.simulation import BioSim
from biosim.landscape import Jungle
from biosim import animals

"""
Compatibility check for BioSim simulations.

This script shall function with biosim packages written for
the INF200 project January 2016.
"""

__author__ = "Hans Ekkehard Plesser, NMBU"
__email__ = "hans.ekkehard.plesser@nmbu.no"


if __name__ == '__main__':
    plt.ion()

    geogr = """OOOOOOOOOOOOOOOOOOOO
               OOOOOOOOSMMMMJJJJJJJ
               OSSSSSJJJJMMJJJJJJJ
               OSSSSSSSSSMMJJJJJJ
               OSSSSSJJJJJJJJJJJ
               OSSSSSJJJDDJJJSJJJ
               OSSJJJJJDDDJJJSSSS
               OOSSSSJJJDDJJJSOOO
               OSSSJJJJJDDJJJJJJ
               OSSSSJJJJDDJJJJOO
               OOSSSSJJJJJJJJOOO
               OOOSSSSJJJJJJJOOO
               OOOOOOOOOOOOOOOOO"""
    geogr = textwrap.dedent(geogr)

    ini_herbs = [{'loc': (10, 10),
                  'pop': [{'species': 'Herbivore',
                           'age': 5,
                           'weight': 20}
                          for _ in xrange(150)]}]
    ini_carns = [{'loc': (10, 10),
                  'pop': [{'species': 'Carnivore',
                           'age': 5,
                           'weight': 20}
                          for _ in xrange(40)]}]

    animals.Herbivore.set_parameters({'zeta': 3.2, 'xi': 1.8})
    animals.Carnivore.set_parameters({'a_half': 70, 'phi_age': 0.5,
                                      'omega': 0.3, 'F': 65,
                                      'DeltaPhiMax': 9.})
    Jungle.set_parameters({'fmax': 700})

    sim = BioSim(island_map=geogr, ini_pop=ini_herbs + ini_carns,
                 seed=123456)
    sim.simulate(num_steps=100, vis_steps=1, img_steps=2000)

    sim.add_population(population=ini_carns)
    sim.simulate(num_steps=100, vis_steps=1, img_steps=2000)
```