Modelling the Ecosystem of Rossumøya

Yngve Mardal Moe
Director, Environmental Protection Agency of Pylandia

Daniel Prince Ngoc Bao Ahmed Albuni
Special Advisors

Version 2019.1: 2019-01-03

Project in Advanced Programming at REALTEK / NMBU, January 2019.

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 is 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, 22 January 2020, 12.00**.

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 January 22nd 2019 deliver source code and documentation to EPAP. Details on how code and documentation are to be delivered will be provided later.

Since EPAP has 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 present their code and illustrative results during interviews held on January 27nd/28th 2020. Details will be given at a later date.

Date	Milestone
06 Jan 2020	Code Repository and PyCharm Project
07 Jan 2020	Problem and requirements analysis.
10 Jan 2020	Demonstration of a first functional simulation of *herbivores* in one place (no migration).
13 Jan 2020	Project plan for remaining work.
17 Jan 2020	Demonstration of a working simulation of herbivores and carnivores, all types of terrain and simple visualization.
22 Jan 2020	**12.00:** Full simulation code with documentation.
25 Jan 2020	**12.00:** PDF and animation for oral presentation.
27 Jan 2020	**09.00–17.00:** Presentation and assessment interviews.
28 Jan 2020	**09.00–17.00:** Presentation and assessment interviews.

Table 1: Project milestones.

2 The Nature of Rossumøya

2.1 Geography

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

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

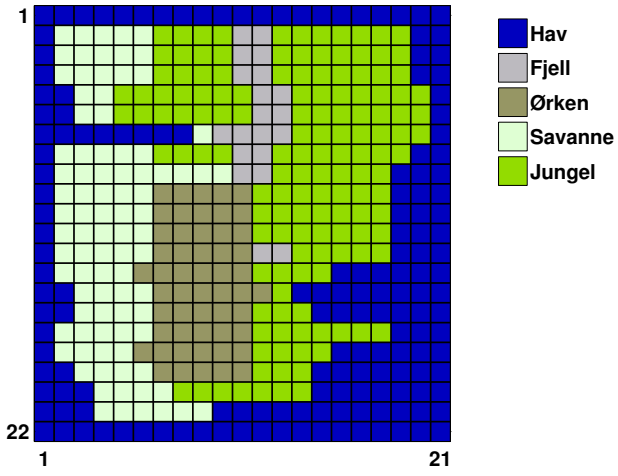

Figure 1: Landscape types on Rossumøya according to the last survey.

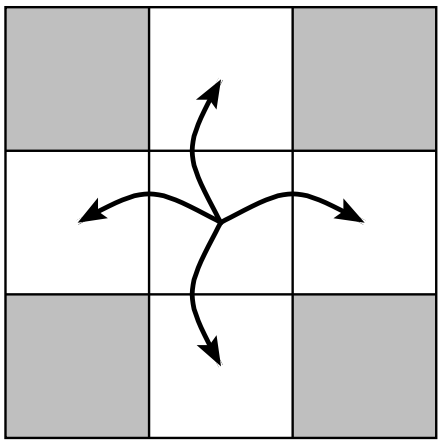

Figure 2: An animal that is in the middle cell can move to one of the four neighboring cells but not to cells diagonally displaced (gray).

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 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)

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

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.

2.1.5 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 $f_{\max}^{\text{Sav}}$.

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

$$\Phi = \begin{cases} 0 & w \leq 0 \\ q^+(a, a_{\frac{1}{2}}, \phi_{\text{age}}) \times q^-(w, w_{\frac{1}{2}}, \phi_{\text{weight}}) & \text{else} \end{cases} \tag{3}$$

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 neigboring 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 is 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 animal 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

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

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

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

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 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

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

  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 $w < \zeta(w_{\text{birth}} + \sigma_{\text{birth}})$.
- Gender plays no role in mating.
- Each animal can give birth to at most one offspring per year.
- At birth, the mother animal loses ξ times the actual birthweight of the baby.

[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.

- If the mother would lose more than her own weight, then no baby is born and the weight of the mother remains unchanged.

6. Death An animal dies

- with certainty if its fitness is $\Phi = 0$;
- with probability

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

otherwise.

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. Each 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 herbivores 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.

2.2.3 Parameters

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 and 3.

[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.

Param.	Herb.	Carn.	Name
w_{birth}	8.0	6.0	`w_birth`
σ_{birth}	1.5	1.0	`sigma_birth`
β	0.9	0.75	`beta`
η	0.05	0.125	`eta`
$a_{\frac{1}{2}}$	40.0	60.0	`a_half`
ϕ_{age}	0.2	0.4	`phi_age`
$w_{\frac{1}{2}}$	10.0	4.0	`w_half`
ϕ_{weight}	0.1	0.4	`phi_weight`
μ	0.25	0.4	`mu`
λ	1.0	1.0	`lambda`
γ	0.2	0.8	`gamma`
ζ	3.5	3.5	`zeta`
ξ	1.2	1.1	`xi`
ω	0.4	0.9	`omega`
F	10.0	50.0	`F`
$\Delta\Phi_{\text{max}}$	—	10.0	`DeltaPhiMax`

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. All parameter values shall be positive (≥ 0), except for $\Delta\Phi_{\text{max}}$, which shall be strictly positive (> 0). Furthemore, $\eta \leq 1$ is required.

Param.	Jungle	Savanne	Name
f_{max}	800.0	300.0	`f_max`
α	—	0.3	`alpha`

Table 3: Example values for parameters of Jungle and Savannah cells. These values are used in the examples shown in the text, but have no special meaning.

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 condition that each animal can migrate at most 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 Guidelines

3.1 Development process

All source code shall be managed on a Git repository hosted on GitHub. 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 the PyTest framework.

3.2 Deliverables

Software A Python package

1. compatible with Python 3.6;
2. installable using standard Python distribution tools;
3. structured according to the supplied project template;
4. organized so that a user can carry out simulations using the methods of a class `BioSim`, which forms the interface for the package, see also Appendix B;
5. written in well-structured, documented, and efficient code following PEP8 guidelines (maximum line length 80 characters);
6. including unit and integration tests covering the code (PyTest framework);
7. including user-level documentation of all modules, classes and public methods generated with Sphinx from docstrings, allowing domain experts to use the software;
8. including working examples;
9. passing a standard test of interface tests provided by EPAP and working with the compatibility check script given in Appendix A.

Presentation Software and exemplary results will be presented in oral presentations. 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 4.

`O`	ocean
`J`	jungle
`S`	savannah
`D`	desert
`M`	mountain

Table 4: Codes for landscape types.

The geography string must consist solely of "O" around the edges and no characters other than the letters shown in Table 4 must occur in the string. The software will raise a `ValueError`' if any requirements on the geography specification is violated.

The coordinate system for the island is as illustrated in Figure 1:

- The upper left corner has coordinates $(0, 0)$.
- Coordinates increase downward and to the right.
- The first coordinate enumerates the rows, the second the columns.

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. 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 2 and 3 shall be used.
2. The software shall have built-in default values for all parameters, so that simulations can be carried out without setting parameters.
3. It shall 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 illegal parameter values, such as negative amounts of fodder.
6. Any errors detected in parameter dictionaries shall raise a `ValueError`.

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 by passing a list[4] to 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}]}]
```

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 placed in cells where animals can stay;

If a placement violates these conditions, the program shall raise a `ValueError`.

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.

Running a simulation twice with the same random seed shall yield identical results.

[4] More precisely, an *iterable*.

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.

3.4.2 Status Information

When the simulation is stopped, it shall be possible to obtain the following information through Python properties of the `BioSim` class with prescribed names:

Year Number of years that have been simulated (property: `year`).

Total number of animals The total number of animals on the island (property: `num_animals`).

Total number of animals by species The total number of herbivores and carnivores, respectively, in a dictionary with keys `'Herbivore'` and `'Carnivore'` (property: `num_animals_per_species`).

Per-cell animal count For each cell, the number of animals shall be available as a Pandas dataframe with columns `Row`, `Col`, `'Herbivore'` and `'Carnivore'` containing the cell coordinates and animal numbers (property: `animal_distribution`).

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 elements:

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, including color bars.

Year The year shall appear in the graphics window.

The user shall be able to specify

- after how many years the graphics are to be updated (the default is each year);
- the limits for the vertical axis in the line graph of animal numbers;

- the limits for the color code in the population maps, separate for both species.

Furthermore, the user shall be able to request that the graphic is saved as file after certain intervals (which must be multiples of the interval for updating the graphics). In this case, the user shall specify the start of the file name of the graphic files and the file-name suffix, which will determine the file type. The files that 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_00000.png`, `bs_00001.png`, `bs_00002.png`, ...

More information about creating movies will be provided later.

4 Further development

This project opens up many opportunities for further development.

Optimization Make the simulation as fast as possible by analyzing the time spent in the program and eliminating bottlenecks. Suggestions for optimization will be given in the January block.

Serialization To examine how different changes affect an ecosystem, it may be useful to save the complete state of the ecosystem to file, so that one can continue the simulation from the saved state with various modifications to the environment. Add code that allows the user to save the complete simulation state to file and to restart the simulation from file.

Parallelization Simulation may be made more efficient using parallel computing for parts of the simulation.

Graphical User Interface Create an interactive GUI so the user can choose specific cells to view information from and modify the simulation parameters mid simulation.

Saving result mid simulation It is often useful to save the simulation results gradually during the simulation. Add the option to write the status information to disc (as a CSV) every iteration.

A Compatibility check

Two compatibility checks are provided by EPAP. The `biosim` package shall pass both:

- `test_biosim_interface.py` contains a set of tests for the interface of the `BioSim` class and is to be used with PyTest.
- `check_sim.py` (see below) is a sample simulation script that shall work with the package you develop. Running the script shall not require any user input.

Both files are available from the project template. The test file shall be included in the `tests` directory in your package, the sample script in the `examples` directory.

No changes shall be made to either file except with express permission of EPAP.

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

import textwrap
import matplotlib.pyplot as plt

from biosim.simulation import BioSim

"""
Compatibility check for BioSim simulations.

This script shall function with biosim packages written for
the INF200 project January 2019.
"""

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


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

    geogr = """\
                OOOOOOOOOOOOOOOOOOOO
                OOOOOOOOSMMMMJJJJJJO
                OSSSSSJJJJMMJJJJJJOO
                OSSSSSSSSSMMJJJJJOOO
                OSSSSSJJJJJJJJJJOOO
                OSSSSSJJJDDJJJSJJOOO
                OSSJJJJJDDDJJJSSSSOOO
                OOSSSSJJJDDJJJSOOOOOO
                OSSSJJJJJDDJJJJJJOOO
                OSSSSJJJJDDJJJJOOOOOO
                OOSSSSJJJJJJJJOOOOOOO
                OOOSSSSJJJJJJJOOOOOOO
                OOOOOOOOOOOOOOOOOOOO"""
    geogr = textwrap.dedent(geogr)

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

    sim = BioSim(island_map=geogr, ini_pop=ini_herbs,
                 seed=123456)

    sim.set_animal_parameters('Herbivore', {'zeta': 3.2, 'xi': 1.8})
    sim.set_animal_parameters('Carnivore', {'a_half': 70, 'phi_age': 0.5,
                                            'omega': 0.3, 'F': 65,
                                            'DeltaPhiMax': 9.})
    sim.set_landscape_parameters('J', {'f_max': 700})

    sim.simulate(num_years=100, vis_years=1, img_years=2000)

    sim.add_population(population=ini_carns)
    sim.simulate(num_years=100, vis_years=1, img_years=2000)

    plt.savefig('check_sim.pdf')

    input('Press ENTER')
```

B BioSim class interface

The BioSim class shall have at least the methods and properties specified below, with the semantics specified by the docstrings.

```
class BioSim:
    def __init__(self, island_map, ini_pop, seed,
                 ymax_animals=None, cmax_animals=None,
                 img_base=None, img_fmt='png'):
        """
        :param island_map: Multi-line string specifying island geography
        :param ini_pop: List of dictionaries specifying initial population
        :param seed: Integer used as random number seed
        :param ymax_animals: Number specifying y-axis limit for graph showing animal numbers
        :param cmax_animals: Dict specifying color-code limits for animal densities
        :param img_base: String with beginning of file name for figures, including path
        :param img_fmt: String with file type for figures, e.g. 'png'

        If ymax_animals is None, the y-axis limit should be adjusted automatically.

        If cmax_animals is None, sensible, fixed default values should be used.
        cmax_animals is a dict mapping species names to numbers, e.g.,
           {'Herbivore': 50, 'Carnivore': 20}

        If img_base is None, no figures are written to file.
        Filenames are formed as

            '{}_{:05d}.{}'.format(img_base, img_no, img_fmt)

        where img_no are consecutive image numbers starting from 0.
        img_base should contain a path and beginning of a file name.
        """

    def set_animal_parameters(self, species, params):
        """
        Set parameters for animal species.

        :param species: String, name of animal species
        :param params: Dict with valid parameter specification for species
        """

    def set_landscape_parameters(self, landscape, params):
        """
        Set parameters for landscape type.

        :param landscape: String, code letter for landscape
        :param params: Dict with valid parameter specification for landscape
        """

    def simulate(self, num_years, vis_years=1, img_years=None):
        """
        Run simulation while visualizing the result.

        :param num_years: number of years to simulate
        :param vis_years: years between visualization updates
        :param img_years: years between visualizations saved to files (default: vis_years)

        Image files will be numbered consecutively.
        """

    def add_population(self, population):
        """
        Add a population to the island

        :param population: List of dictionaries specifying population
        """

    @property
    def year(self):
        """Last year simulated."""

    @property
    def num_animals(self):
        """Total number of animals on island."""

    @property
    def num_animals_per_species(self):
        """Number of animals per species in island, as dictionary."""

    @property
    def animal_distribution(self):
        """Pandas DataFrame with animal count per species for each cell on island."""

    def make_movie(self):
        """Create MPEG4 movie from visualization images saved."""
```