Norwegian University of Life Sciences

A PROJECT IN ADVANCED PROGRAMMING AT REALTEK, NMBU, JANUARY 2023.

Modelling the Ecosystem of Rossumøya

This Project is completed under the guidance of :

Hans Ekkehard Plesser

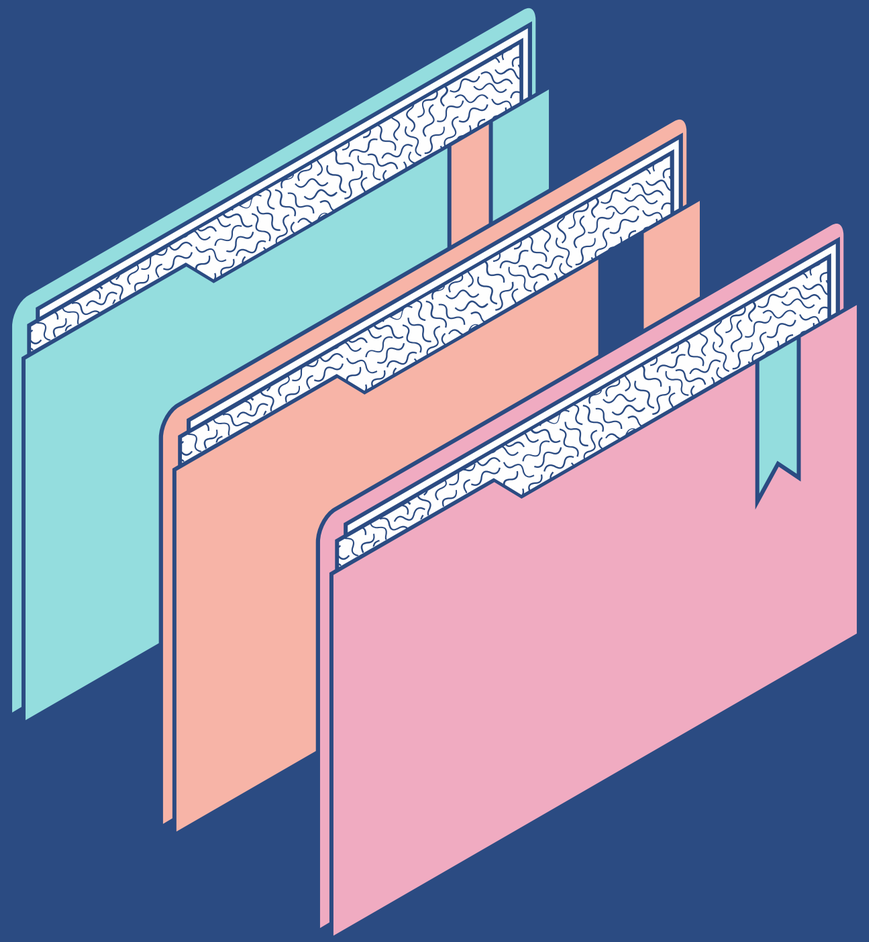
DIRECTOR, ENVIRONMENTAL PROTECTION AGENCY OF PYLANDIA

BIOSIM A36 NAVNEET SUSHANT

Team - A36

Navneet Sharma & Sushant Kumar Srivastava

Norwegian University of Life Sciences



Project Objective

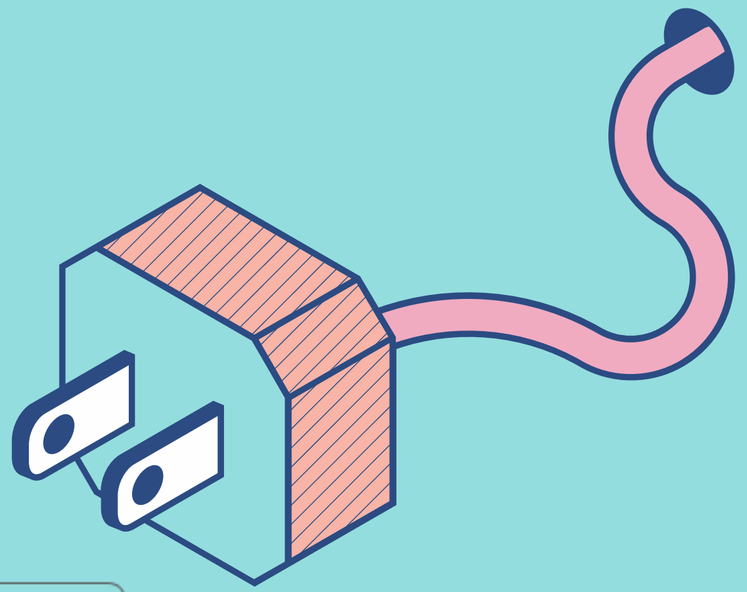
TO SIMULATE THE POPULATION DYNAMICS OF FAUNA ON ROSSUMØYA ISLAND

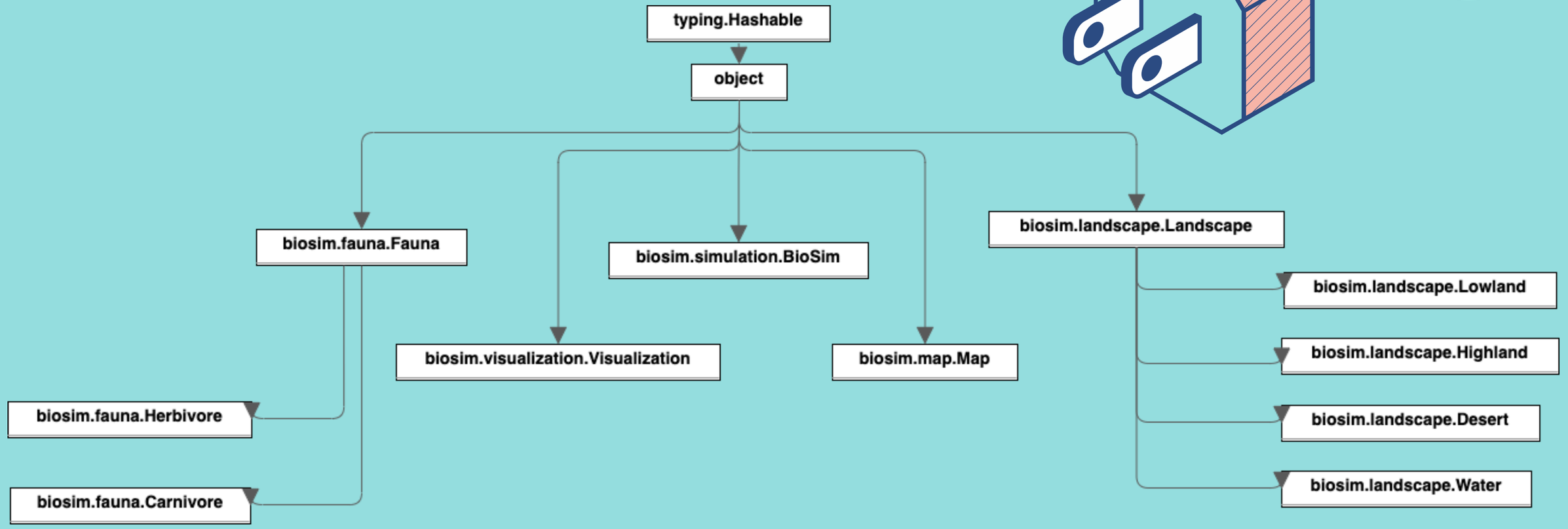
- Two fauna classes namely - **Herbivores and Carnivores**
- Island Landscape types namely - **Highland, Lowland, Desert and Water**
- Seasons on Rossumøya : **Procreation, Feeding, Migration, Aging, Loss of weight n and Death**
- Simulation show using graphics (visualization)

Norwegian University of Life Sciences

Solution Implemented

We used Python with Object Oriented Programming Approach to solve this goal for the project.





Class methods

BELOW ARE THE SNIPPETS OF ALL THE FUNCTIONS USED INSIDE VARIOUS CLASSES

Source code and Documentation

All functions used in src files are available here,they are arranged in a hierarchical order.
Browse them in order ,if you like :)

- Simulation
  - The simulation module
    - BioSim
      - BioSim.add_population()
      - BioSim.age_animals_per_species()
      - BioSim.fitness_animals_per_species()
      - BioSim.make_movie()
      - BioSim.num_animals
      - BioSim.num_animals_per_species
      - BioSim.set_animal_parameters()
      - BioSim.set_landscape_parameters()
      - BioSim.simulate()
      - BioSim.weight_animals_per_species()
      - BioSim.year
- Landscape
  - The landscape module
    - Desert
      - Desert.fodder_grow_and_feeding()
    - Highland
      - Highland.fodder_grow_and_feeding()
    - Landscape
      - Landscape.add_migrated_population()
      - Landscape.add_newborn()
      - Landscape.age_increase()
      - Landscape.animal_die()
      - Landscape.animal_migrate()
      - Landscape.feed_carnivore()
      - Landscape.feed_herbivore()
      - Landscape.reset_animals()
      - Landscape.set_parameters()
      - Landscape.verify_non_valid_parameters()
      - Landscape.verify_parameters()
      - Landscape.weight_decrease()
    - Lowland
      - Lowland.fodder_grow_and_feeding()
    - Water

- Map
  - The map module
    - Map
      - Map.add_population()
      - Map.calculate_animal_count()
      - Map.check_dict_type()
      - Map.check_invalid_map()
      - Map.create_cells()
      - Map.create_neighbours_dict()
      - Map.geo_list()
      - Map.get_pop_age_carn()
      - Map.get_pop_age_herb()
      - Map.get_pop_fitness_carn()
      - Map.get_pop_fitness_herb()
      - Map.get_pop_matrix_carn()
      - Map.get_pop_matrix_herb()
      - Map.get_pop_tot_num()
      - Map.get_pop_tot_num_carn()
      - Map.get_pop_tot_num_herb()
      - Map.get_pop_weight_carn()
      - Map.get_pop_weight_herb()
      - Map.livable_cell_calculate()
      - Map.migrate_cell_calculate()
      - Map.set_parameters()
      - Map.unique_columns()
      - Map.unique_rows()
      - Map.yearly_cycle()
- Fauna
  - The fauna module
    - Carnivore
    - Fauna
      - Fauna.age_increase()
      - Fauna.birth_prob()
      - Fauna.calculate_fitness()
      - Fauna.check_not_defined_params()
      - Fauna.die()
      - Fauna.die_prob()
      - Fauna.kill_prob()
      - Fauna.move_prob()
      - Fauna.set_parameters()
      - Fauna.weight_decrease()
      - Fauna.weight_decrease_on_birth()
      - Fauna.weight_default()
      - Fauna.weight_increase_on_eat()
    - Herbivore

- Visualization
  - The visualization module
    - Visualization
      - Visualization.draw_animal_count_plot()
      - Visualization.draw_frequency_graphs()
      - Visualization.draw_heatmap()
      - Visualization.draw_layout()
      - Visualization.draw_map()
      - Visualization.save_graphics()
      - Visualization.update_animal_count()
      - Visualization.update_frequency_graphs()
      - Visualization.update_heatmap()
      - Visualization.update_plot()

Norwegian University of Life Sciences

Rossumøya's yearly cycle

1 — 2 — 3 — 4 — 5 — 6

STEP

Add Newborn/ Procreation

The season first of the year starts with the birth of newborns

STEP

Grow Fodder and feeding

The herbivores eat fodder from Highlands or Lowlands while carnivores can feed on herbivores in lowland, highland or desert

STEP

Animal Migration

Fauna classes have the ability to migrate with some migration probability and we used a flag - 'has_migrated' to manage the process optimally

STEP

Age Increase

Every fauna has to grow old at the end of the year.

STEP

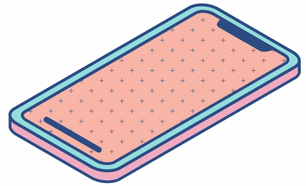
Weight Decrease

At the end of each year, every animal losses some weight and this also affects its fitness so we do the fitness calculation and store it to use in the next year for other calculations.

STEP

Animal Die

The death of the animal is determined using the death probability and this marks
as the last step in the season.



Norwegian University of Life Sciences

First approach

- Iterating through all the landscape cells in one cycle. But later realized that this will cause big problems after migration

```python
def yearly_cycle(self):
    """This method calls, in order, the methods that compound
    the yearly cycle dynamics of the island, such that:

        1. Animal's birth;
        2. Animal's feeding;
        3. Animal's migration;
        4. Animal's migration;
        5. Animal's aging;
        6. Animal's weight loss;
        7. Animal's death.
    """
    for loc, loc_object in self.livable_cell_calculate().items():
        loc_object.add_newborn()
        loc_object.fodder_grow_and_feeding()
        loc_object.animal_migrate(self.neighbours_dict[loc])
        loc_object.add_migrated_population()
        loc_object.age_increase()
        loc_object.weight_decrease()
        loc_object.animal_die()
```

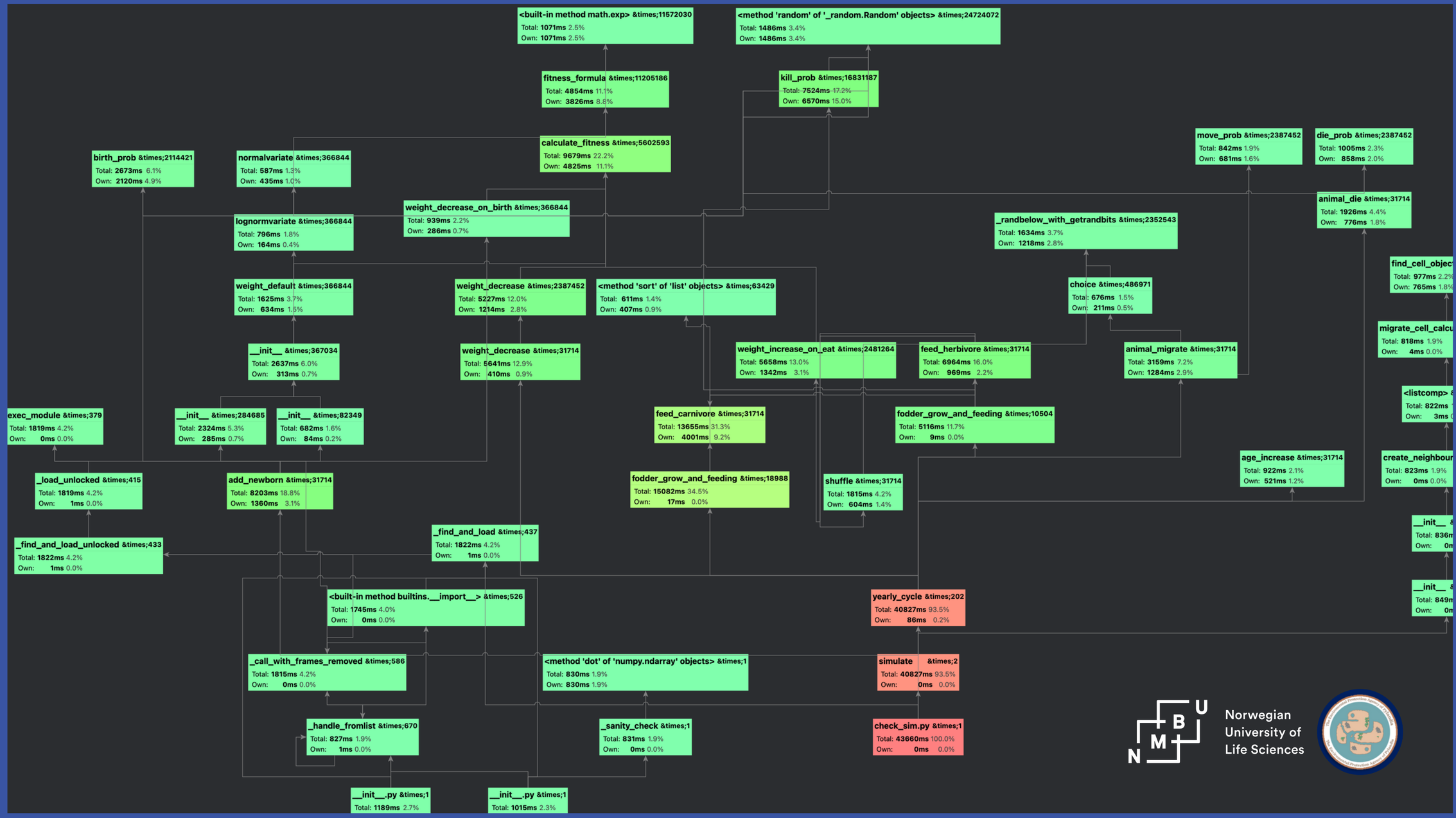
Optimized Approach

- Iterating through all the landscape cells in two cycles. Second cycle after the migration to do ageing, weight loss and death for animals who migrated to new locations

```python
def yearly_cycle(self):
    """This method calls, in order, the methods that compound
    the yearly cycle dynamics of the island, such that:

        1. Animal's birth;
        2. Animal's feeding;
        3. Animal's migration;
        4. Animal's migration;
        5. Animal's aging;
        6. Animal's weight loss;
        7. Animal's death.
    """
    for loc, loc_object in self.livable_cell_calculate().items():
        loc_object.add_newborn()
        loc_object.fodder_grow_and_feeding()
        loc_object.animal_migrate(self.neighbours_dict[loc])
    for loc, loc_object in self.livable_cell_calculate().items():
        loc_object.add_migrated_population()
        loc_object.age_increase()
        loc_object.weight_decrease()
        loc_object.animal_die()
```

Profiling Results : Call Graph



Norwegian University of Life Sciences

Before optimization

Norwegian University of Life Sciences

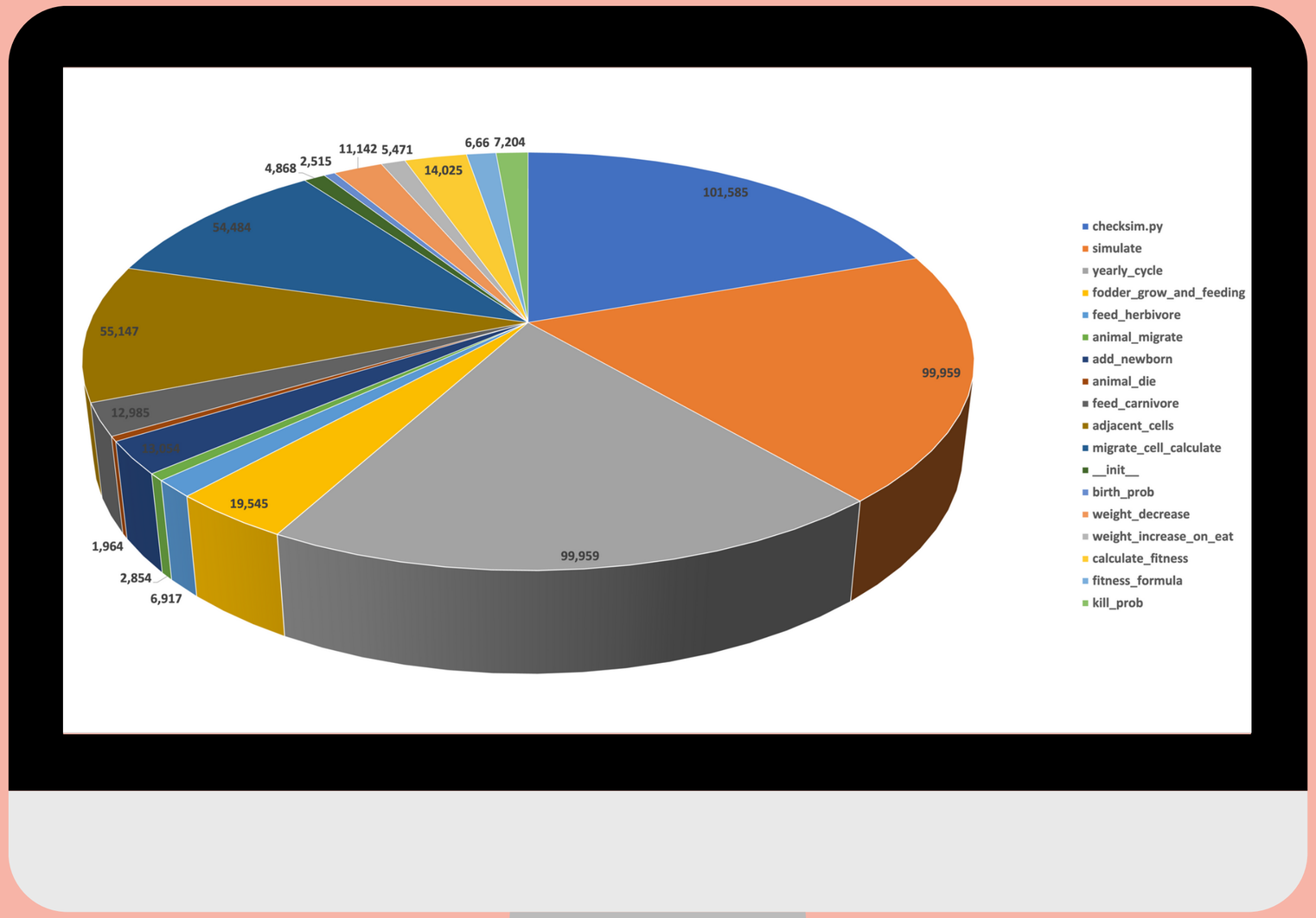
TOTAL TIME TAKEN: 101 SEC

CALL TAKING HIGHEST TIME THAT CAN BE OPTIMIZED:

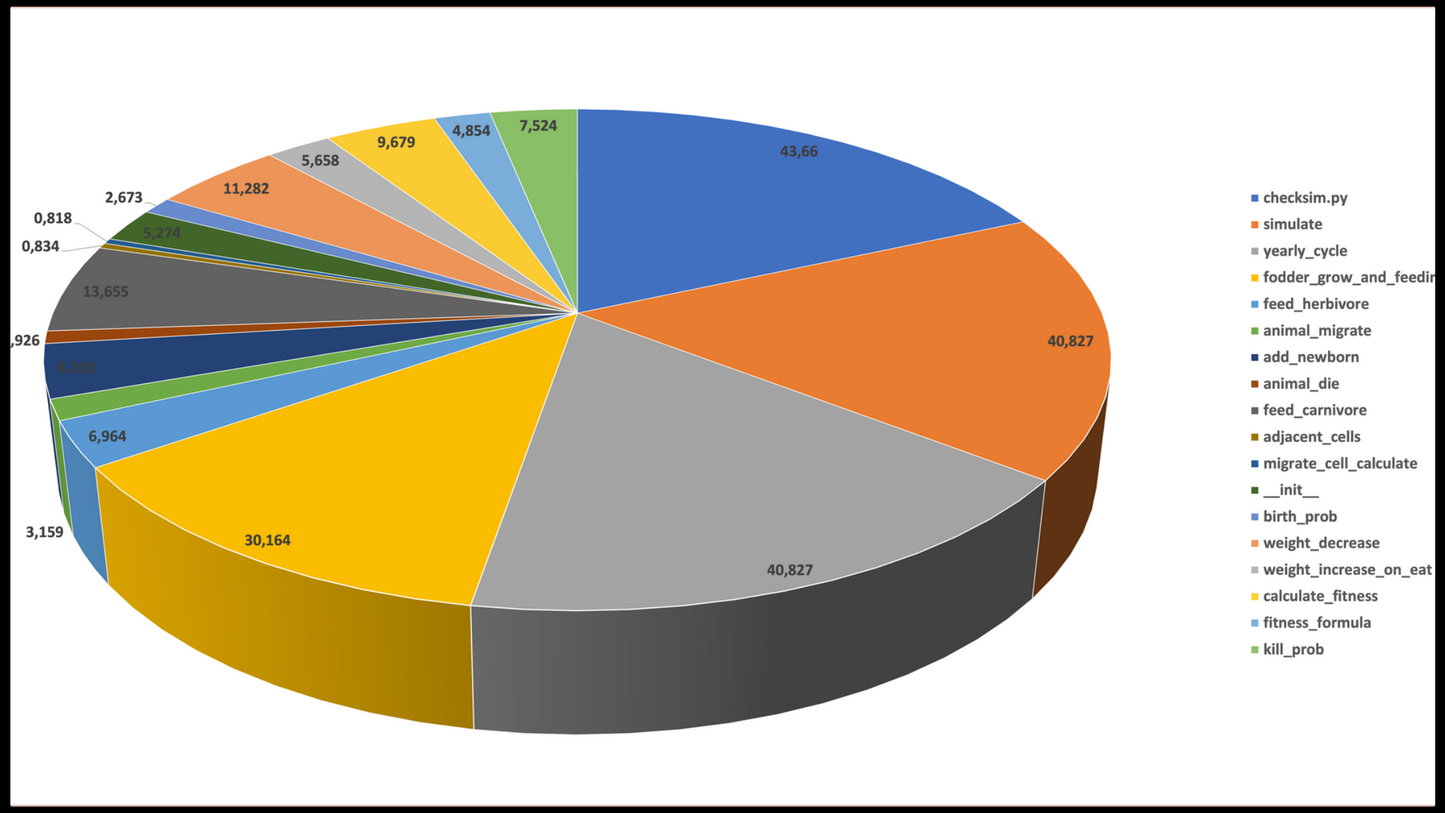
- ADJACENT_CELLS - 55,147
- CALCULATE_FITNESS - 14,025

OPTIMIZATION ACTIONS

- CALCULATION OF NEIGHBOUR CELLS
- CALCULATION OF FITNESS
- USING UPDATES IN PLACE FOR VISUALIZATION



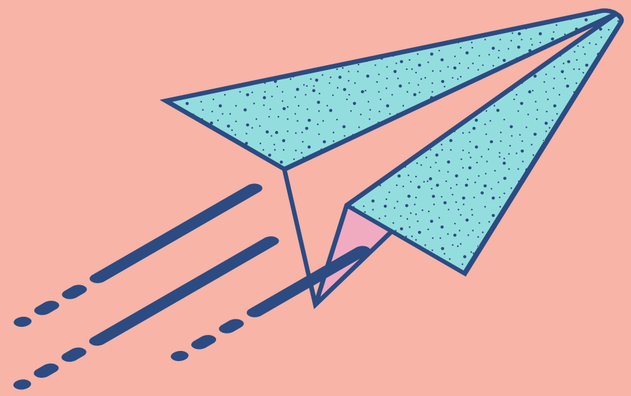
After optimization



TOTAL TIME TAKEN: 43,6 SEC

OPTIMIZED CALL TIMINGS:

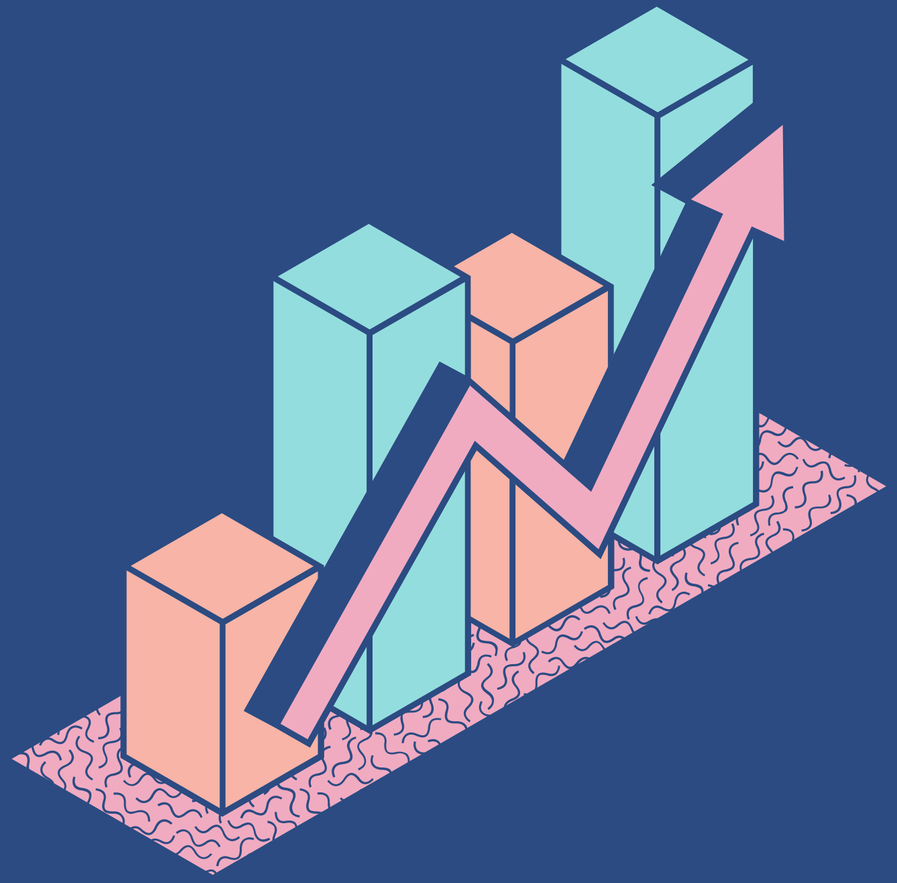
- ADJACENT_CELLS - 0,834
- CALCULATE_FITNESS - 9,67

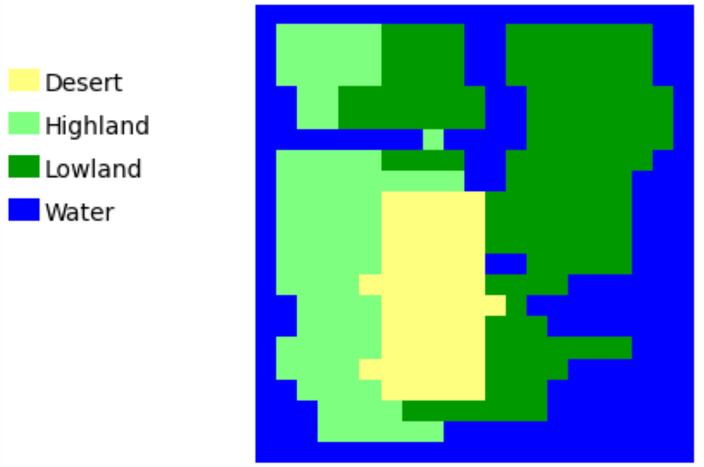


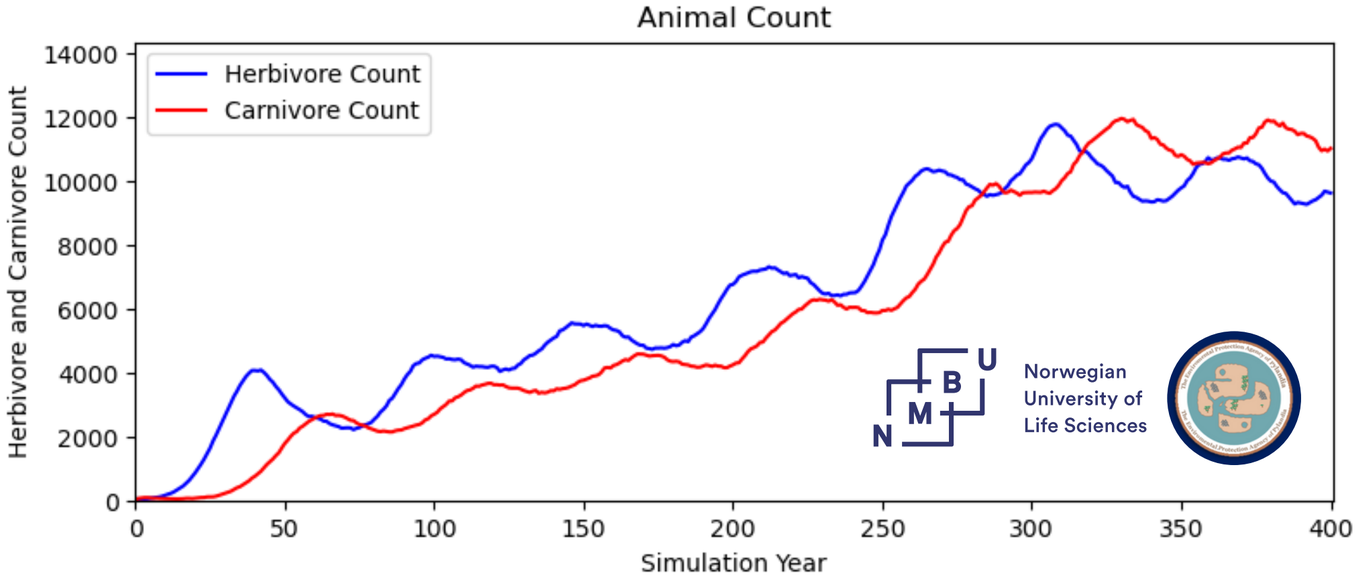
Norwegian University of Life Sciences

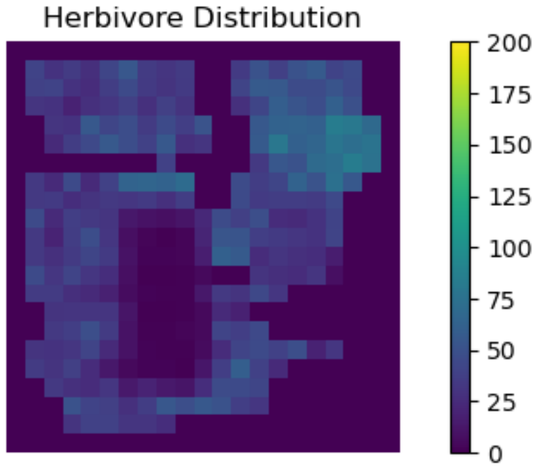
Visualization of the simulation

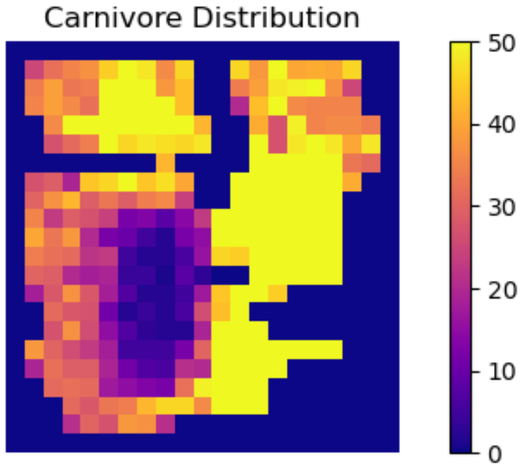
Update in place logics have been implemented to make the visuals and save graphics faster



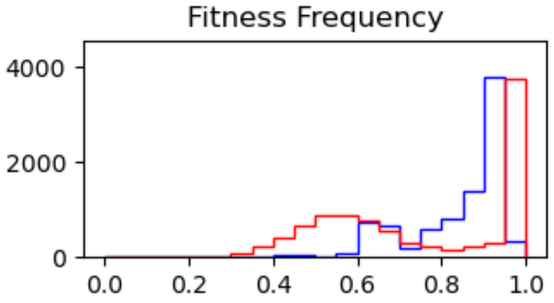

Desert
Highland
Lowland
Water

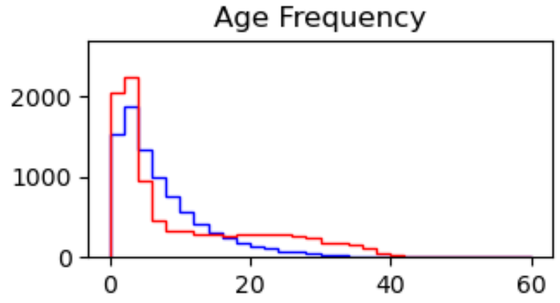

Animal Count
Herbivore Count
Carnivore Count
Herbivore and Carnivore Count
Simulation Year
0
2000
4000
6000
8000
10000
12000
14000
0
50
100
150
200
250
300
350
400
Norwegian University of Life Sciences

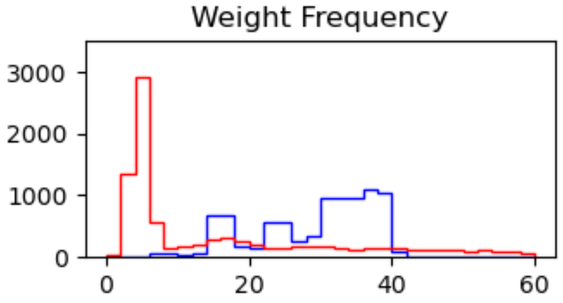

Herbivore Distribution
200
175
150
125
100
75
50
25
0


Carnivore Distribution
50
40
30
20
10
0

Year: 400


Fitness Frequency
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0


Age Frequency
2000
1000
0
0
20
40
60


Weight Frequency
3000
2000
1000
0
0
20
40
60

Learnings from the project

HOW THIS PROJECT HELPED US LEARN

- We learned how to make a project in test-driven development
- By making a hierchical structure of our code, we learned how to plan time effectively and divide the work among ourselves
- We learned how to optimize the code for more efficient simulation
- By creating testing files before creating the methods helped us to think more about the debugging side of the project which helped us a lot with just few bugs
- We learned how to display the results using graphics through the means of line graphs, heatmaps and histograms
- With tox and flake8, we learned how to check the style and quality of our code

We are happy to do discussions at length now!

Norwegian University of Life Sciences

Thank You!

Tusen takk !

“We need technology in every classroom and in every student and teacher’s hand, because it is the pen and paper of our time, and it is the lens through which we experience much of our world."

DAVID WARLICK