Ntegano Bahenda Yvon Dylan - 515657
Pietrasanta Sebastiano - 513054

Planisuss

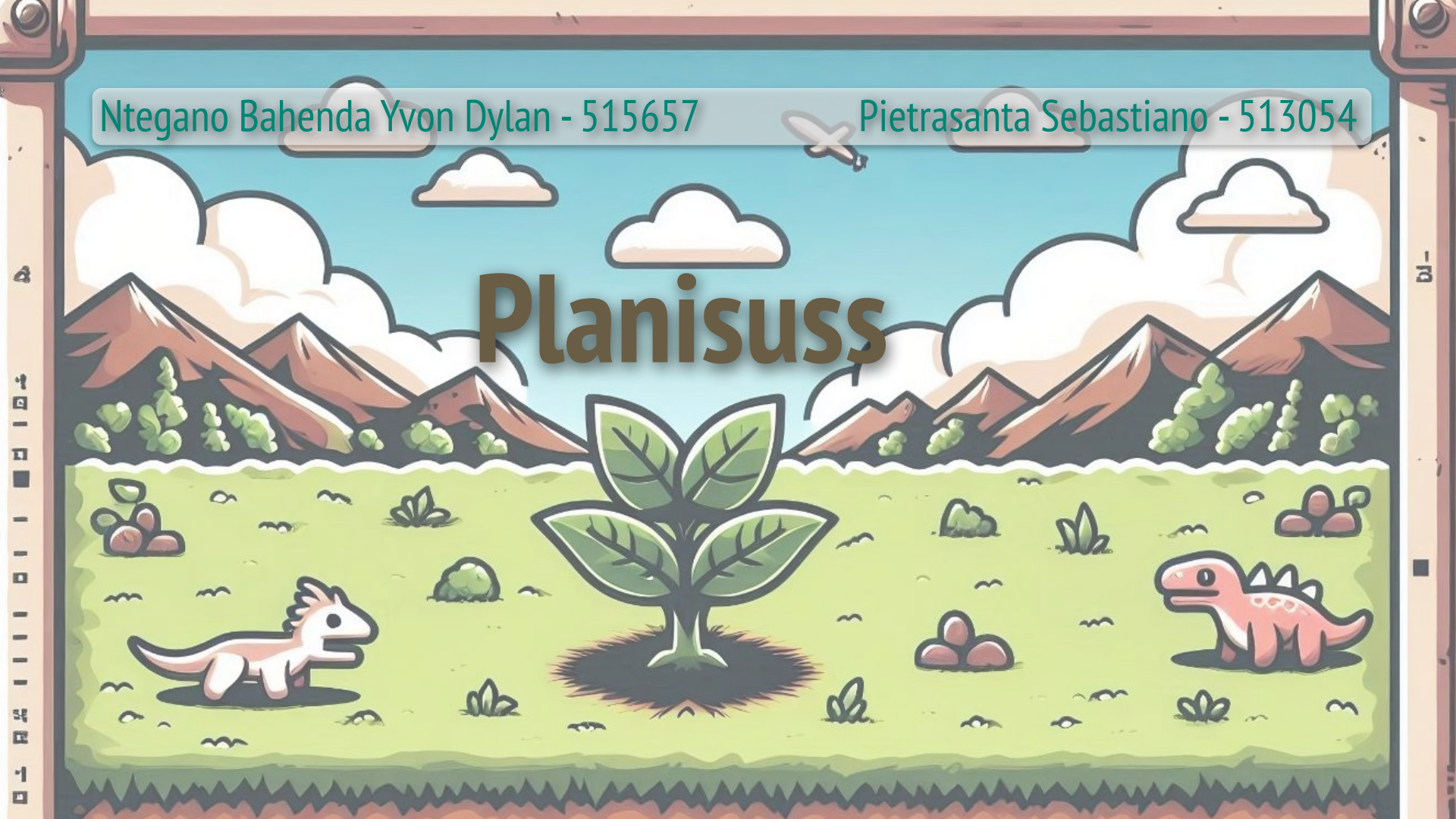


Table of contents

Planisuss proverb

Every morning in Planisuss, a Erbast wakes up. It knows it must outrun the fastest Carviz or it will be killed.

Every morning in Planisuss, a Carviz wakes up. It knows it must run faster than the slowest Erbast, or it will starve.

It doesn't matter whether you're the Carviz or a Erbast—when the sun comes up, you'd better be running.

Unless you are a Vegetob.

Project Goals and Constraints

Primary Goals:

- Simulate a pseudo-naturalistic world on a grid
- Model interactions among three species

Constraints:

- Balance energy consumption and gain.
- Ensure realistic without complete dominance of one species.
- Clear visualization and interactive simulation.

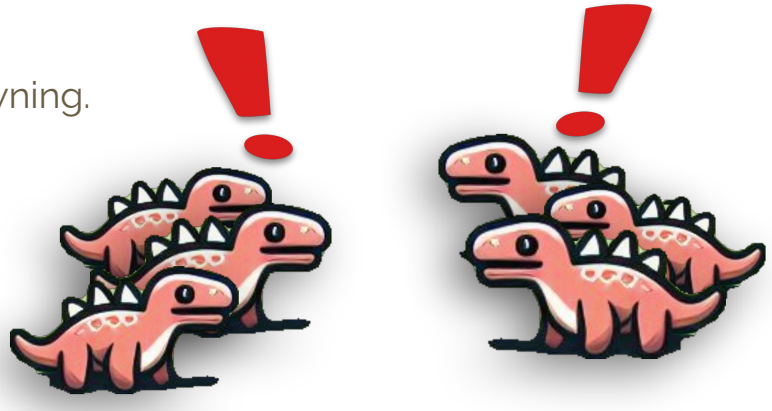
Assumptions, Methods, and Procedures

Design Choices:

- Grid-based spatial design.
- Time represented in discrete steps (days).
- Use of constants for flexibility and experimentation.

Simulation Phases:

- Growth, Movement, Grazing, Struggle, Spawning.



Logical Structure of the Solution

Grid Initialization:

- Assign water and land cells.
- Populate with Vegetob, Erbast, and Carviz.

Each day we loop through the simulation phases.

Implementation Details

Class Definitions:

- `Cell` class for cell properties and actions.
- `Erbast` and `Carviz` classes for species behavior and lifecycle.

Functions:

- `grow`, `move_erbast`, `move_carviz`, `graze`, `hunt`, `spawn`, `fight`, `Create_world`, `Initialize_world`, `get_neighborhood_cells`, `simulate_day`.

```
def hunt(self):
    """
    Simulate the hunt behavior of Carvizes in the cell.
    """
    strongest_erbast = max(self.erbasts, key=lambda erbast: erbast.energy)
    hunt_success = random.random() < 0.3
    if hunt_success:
        prey_energy = strongest_erbast.energy
        lowest_energy_carviz = min(self.carvizes, key=lambda carviz:
carviz.energy)
        for carviz in self.carvizes:
            carviz.energy += prey_energy // len(self.carvizes)
            if carviz == lowest_energy_carviz:
                carviz.energy += prey_energy % len(self.carvizes)
            carviz.social_attitude += 0.05
        self.erbasts.remove(strongest_erbast)
        for erbast in self.erbasts:
            erbast.social_attitude += 0.05
    else:
        for carviz in self.carvizes:
            carviz.social_attitude -= 0.05
```

```
def get_neighborhood_cells(cell):

    adjacent_non_water_cells = []

    for i in range(-const.NEIGHBORHOOD, const.NEIGHBORHOOD + 1):
        for j in range(-const.NEIGHBORHOOD, const.NEIGHBORHOOD + 1):
            if i == 0 and j == 0:
                continue  # Skip the current cell
            neighbor_coords = (cell.coords[0] + i, cell.coords[1] + j)
            neighbor_cell = world[neighbor_coords[0]][neighbor_coords[1]]
            if neighbor_cell and not neighbor_cell.is_water:
                adjacent_non_water_cells.append(neighbor_cell)

    return adjacent_non_water_cells
```

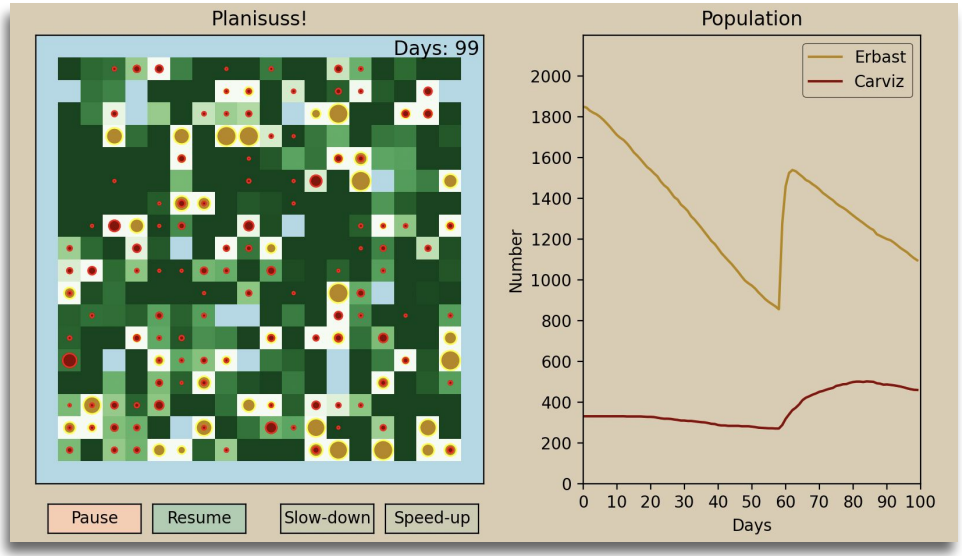
Visualization

Setup:

- World grid and population graphs.
- Interactive controls for animation.

Key Functions:

- `get_vegetation_map`, `update`, interactive buttons for speed control.



Testing and Results

Testing Approach:

- Various scenarios with different initial conditions.
- Evaluated performance and accuracy through visual outputs.

Outcomes:

- Accurate representation of species behavior and interactions.
- Consistent population trends and spatial distributions.

Possible Improvements

Enhancements:

- Detailed and interactive visualizations.
- Code optimization for better performance.
- Addition of complex behaviors and decision-making processes for species.

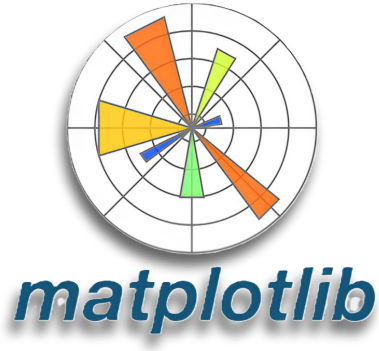
Conclusion

Summary:

- Successful simulation of a fictional ecosystem.
- Clear visualization of population dynamics.

Future Work:

- Implement suggested improvements.
- Explore additional features and scenarios.



Tools and Resources Used

Libraries:

- Matplotlib 3.4.3
- NumPy 1.21.2

Reasons:

- Matplotlib for detailed and interactive plots.
- NumPy for efficient calculations and data manipulation.


Thank you!!