An island simulation

August Steinset and Sunniva Steiro

The code

- Object oriented
- Close collaboration
  - Minimize errors
  - Extensive knowledge
- Testing and documentation
- An aspect of the code: migration

```python
def all_migrate_herb(self):
    """Make the herbivores migrate"""
    migrated_herb = []
    for i, a in enumerate(self.island):
        temp_herb = []
        for j in range(len(a)):
            if i != 0 and j != 0 and i != len(self.island) - 1 and j != len(a) - 1:
                legal_moves = [self.island[i - 1][j].movable, self.island[i + 1][j].movable,
                               self.island[i][j + 1].movable, self.island[i][j - 1].movable]
                temp_herb.append(self.island[i][j].migration_herb(legal_moves))

            else:
                temp_herb.append([[], [], [], []])
        migrated_herb.append(temp_herb)
    for i, a in enumerate(migrated_herb):
        for j, b in enumerate(a):
            if i != 0 and j != 0 and i != len(self.island) - 1 and j != len(a) - 1:
                self.island[i - 1][j].herbivores_on_tile.extend(b[0])
                self.island[i + 1][j].herbivores_on_tile.extend(b[1])
                self.island[i][j + 1].herbivores_on_tile.extend(b[2])
                self.island[i][j - 1].herbivores_on_tile.extend(b[3])
```

```
def migration_herb(self, legal_moves):
    """
    Returns list of all herbivores on tile that will move, in lists of where they will move.

    :param legal_moves: list of boolean values indicating what neighboring
     tiles are available for immigration.
    """
    up = []
    down = []
    left = []
    right = []
    migrate_list = [up, down, right, left]
    remaining_herbivores = []
    for k in range(len(self.herbivores_on_tile)):
        if self.herbivores_on_tile[k].check_migration():
            index = random.randrange(len(legal_moves))
            if legal_moves[index]:
                migrate_list[index].append(self.herbivores_on_tile[k])
            else:
                remaining_herbivores.append(self.herbivores_on_tile[k])
        else:
            remaining_herbivores.append(self.herbivores_on_tile[k])
    self.herbivores_on_tile = remaining_herbivores
    return migrate_list
```

```
def check_migration(self):
    """Finds out if the animal will try to migrate

    :return: True if the animal tries to migrate
    :rtype: bool
    """
    if self.parameter['mu'] * self.fitness > random.random():
        return True
    else:
        return False
```

The code

- Accessibility
- Speed

Simulation example

- Volcanic winter