Project Report : Garden Simulation

Name: Chanuka Dilusha Athalage

Student Number: 21953004

Date: 4/5/2024

Overview

This simulation explores the dynamic interaction among puppies, humans, and squirrels within a residential area filled with a lively ecosystem featuring a house with a spacious garden on a rainy day. In particular, puppies bring curiosity to the simulation by exploring their surroundings in a playful manner, such as getting fed and pampered by humans (owners), barking at strangers, playing with balls, reproducing, and chasing squirrels. Meanwhile, squirrels will try to roam around the garden while avoiding puppies and humans, and human residents provide care to the puppies. The complex and interesting ecology is represented in 2D graphics, and to make things more interesting, different terrains such as land, plants, and a road are used. Each creature goes through a life cycle, and this simulation runs for a day. At night, all the creatures will go back to their homes.

User Guide

- **Setup:** Ensure python3 is installed along with **numpy**, **random** and **matplotlib** libraries.
- **Code usage:** The simulation initialises after executing the Python script which has the main method in it using the command line. The script accepts several inputs from the user to start the simulation.
    - **time_steps** – This variable takes the number of times the user wants to run the simulation.
    - **no_puppies** – This variable takes the number of puppies the user wants.
    - **no_squirrels** – This variable takes the number of squirrels the user wants.
    - **no_humans** – This variable takes the number of humans the user wants.
    - **no_balls** – This variable takes the number of balls the user wants.

Traceability Matrix

Feature	Code Reference	Test Reference	Status	Date Completed
1.0 Animals (example)	Class Animal in creatures.py	Assessment.py		
1.1 Puppy	Class Puppy in creatures.py	Assessment.py		4/5/24
1.1.1 Puppies are plotted as a dog's face	plot_me() method in Puppy class	Assessment.py – plotting puppies	P	1/5/24
1.1.2 Puppies changing their movements	step_change() method in Puppy class	Assessment.py – movement test section	P	1/5/24
1.1.3 Puppies avoiding walls and trees	check_borders() method in Puppy class	Assessment.py – Avoiding obstacles test section	P	1/5/24
1.1.4 Puppies checking distance to others	distance_to_objects() method in Puppy class	Assessment.py – Calculation to other objects test section	P	4/5/24
1.1.5 Puppies reacting to different humans (owners and strangers)	near_human() method in Puppy class	Assessment.py – Reacting to humans test section	P	4/5/24
1.1.6 Puppies reproducing	reproduce_when_near_puppy() method in Puppy class	Assessment.py – Reproducing test section	P	3/5/24
1.1.7 Puppies avoiding tree collision	stop_tree_collision() method in Puppy class	Assessment.py – Tree collision test section	P	3/5/24
1.1.8 Puppies barking at strangers	bark() method in Puppy class	Assessment.py – Barking at strangers test section	P	4/5/24
1.1.9 Puppies getting old	is_old() method in Puppy class	Assessment.py – Puppy aging test section	P	3/5/24
1.1.10 Puppies energy	energy_fading() method in Puppy class	Assessment.py – Energy	P	4/5/24

		decrease testing section		
1.2 Squirrel	Class Squirrel in creatures.py	Assessment.py		4/2/24
1.2.1 Squirrels are plotted as a squirrel's face	plot_me() method in Squirrel class	Assessment.py – plotting Squirrels	P	1/5/24
1.2.2 Squirrels changing their movements	step_change() method in Squirrel class	Assessment.py – movement test section	P	1/5/24
1.2.3 Squirrels avoiding walls	check_borders() method in Squirrel class	Assessment.py – Avoiding obstacles test section	P	1/5/24
1.2.4 Squirrels checking distance to others	distance_to_objects() method in Squirrel class	Assessment.py – Calculation to other objects test section	P	4/5/24
1.3 Human	Class Human in creatures.py	Assessment.py		4/2/24
1.3.1 Humans are plotted as pink circles	plot_me() method in Human class	Assessment.py – plotting Humans	P	1/5/24
1.3.2 Humans changing their movements	step_change() method in Human class	Assessment.py – movement test section	P	1/5/24
1.3.3 Humans avoiding walls	check_borders() method in Human class	Assessment.py – Avoiding obstacles test section	P	1/5/24
1.3.4 Humans checking distance to others	distance_to_objects() method in Human class	Assessment.py – Calculation to other objects test section	P	4/5/24
1.3.5 Humans avoiding tree collision	stop_tree_collision() method in Human class	Assessment.py – Tree collision test section	P	3/5/24
1.4 Ball	Class Ball in creatures.py	Assessment.py		4/2/24
1.4.1 Balls are plotted as green circles	plot_me() method in Ball class	Assessment.py – plotting Balls	P	1/5/24

Discussion

1. Puppy:

   1.1. **plot_me() -** This method takes axes and the size of the yard as parameters. Then it uses the patch module from matplotlib to design the puppy.

   1.2. **step_change() -** This method takes a list of objects as the parameter, namely objects of humans, squirrels, and balls. The method defines the valid moves for the puppy. It uses a foreach loop to iterate through all the objects passed to the function. If the object is within a 10-unit radius, the method calculates the distance to that object and assigns it to the 'move_towards_object' variable for the puppy's next move.

   1.3. **check_borders() -** This method takes the value of the next movement as the parameter. Then it checks if the puppy collides with the borders of the simulator (walls).

   1.4. **distance_to_objects()** – This method takes an object as the parameter. Then it calculates the distance between self.pos and the object position using the Euclidean equation.

   1.5. **near_human()** – This method takes a list of humans, axes and size of the 2D graph as parameters. Then it uses a foreach loop to iterate through humans list and check their human_type. If the human_type is 'owner', it increases the energy level of the puppy (human feeds puppy). If the human_type is 'stranger', it calls the bark() method (to bark at strangers).

   1.6. **reproduce_when_near_puppy()** – This method takes a list of puppies as the parameter. Then it uses a foreach loop to iterate through the puppies list and check if 2 puppies are close together.

   1.7. **stop_tree_collision() –** This method takes the value of the next move of the puppy as the parameter. Then it checks if the puppy is colliding with trees. If the puppy collides with trees, this method changes the moving direction of the puppy.

   1.8. **bark()** – This method takes axes and size of the yard as parameters. Then it annotates a 'woof' near the puppy.

   1.9. **is_old() –** This method is used to check the age of the puppy and increase its age.

   1.10. **energy_fading() –** This method decreases puppy energy in each timestep.

2. Squirrel:
    2.1. **plot_me() -** This method takes axes and the size of the yard as parameters. Then it uses the patch module from matplotlib to design the squirrel.

    2.2. **step_change() -** This method takes a list of objects as the parameter, namely objects of humans and puppies. The method defines the valid moves for the squirrel. It uses a foreach loop to iterate through all the objects passed to the function. If the object is within a 10-unit radius, the method makes the squirrel move fast by increasing the moving speed.

    2.3. **check_borders() -** This method takes the value of the next movement as the parameter. Then it checks if the squirrel collides with walls. If so, it changes the movement of the squirrel.

    2.4. **distance_to_objects()** – This method takes an object as the parameter. Then it calculates the distance between self.pos and the object position using the Euclidean equation.

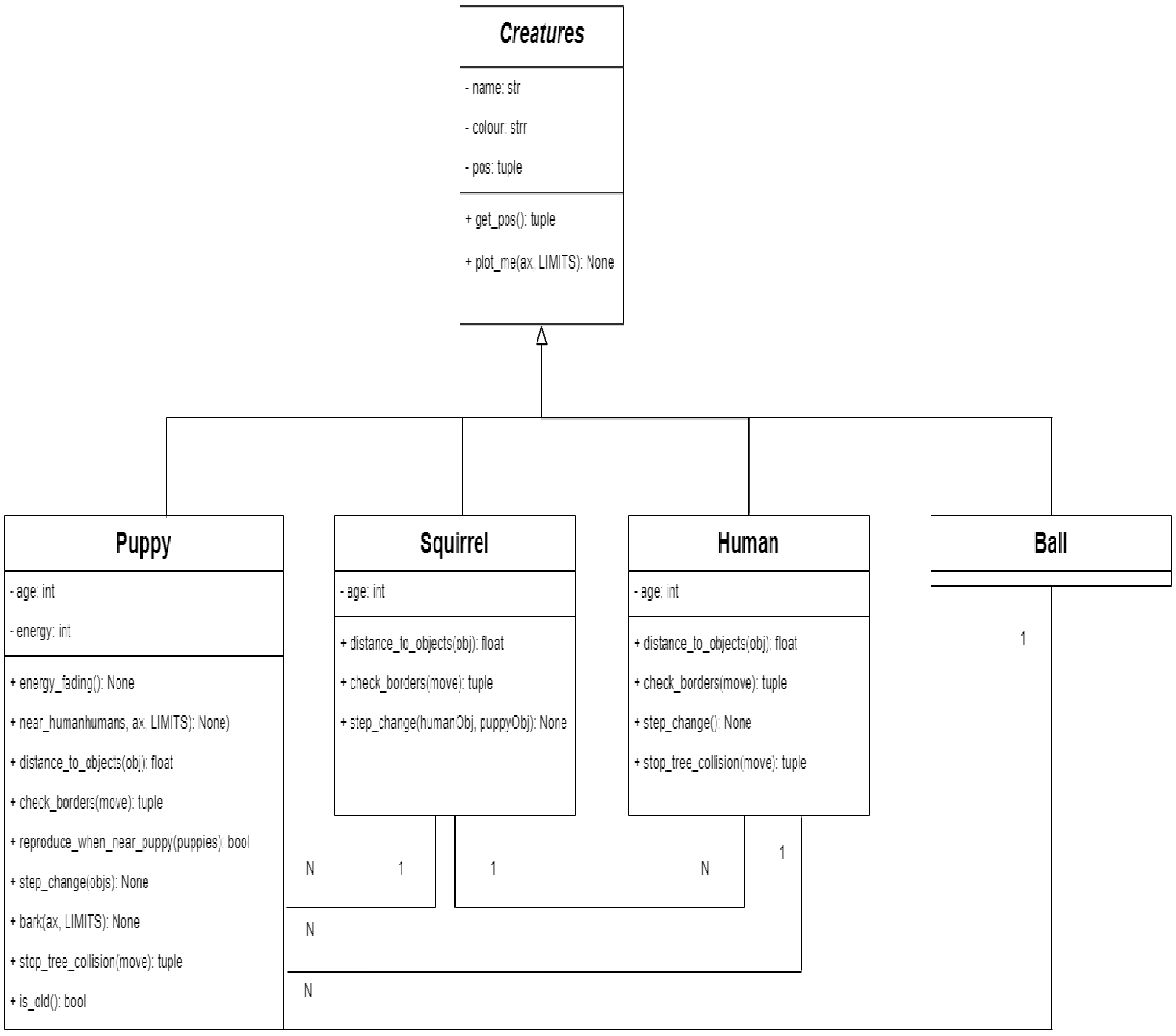
3. Human:
    3.1. **plot_me() -** This method takes axes and the size of the yard as parameters. Then it uses the patch module from matplotlib to design the human.

    3.2. **step_change() -** This method is used to define valid human moves.

    3.3. **check_borders() -** This method takes the value of the next movement as the parameter. Then it checks if the human collides with walls. If so, it changes the movement of the human.

    3.4. **distance_to_objects()** – This method takes an object as the parameter. Then it calculates the distance between self.pos and the object position using the Euclidean equation.

    3.5. **stop_tree_collision()** - This method takes the value of the next move of the human as the parameter. Then it checks if the human is colliding with trees. If the human collides with trees, this method changes the moving direction of the human.

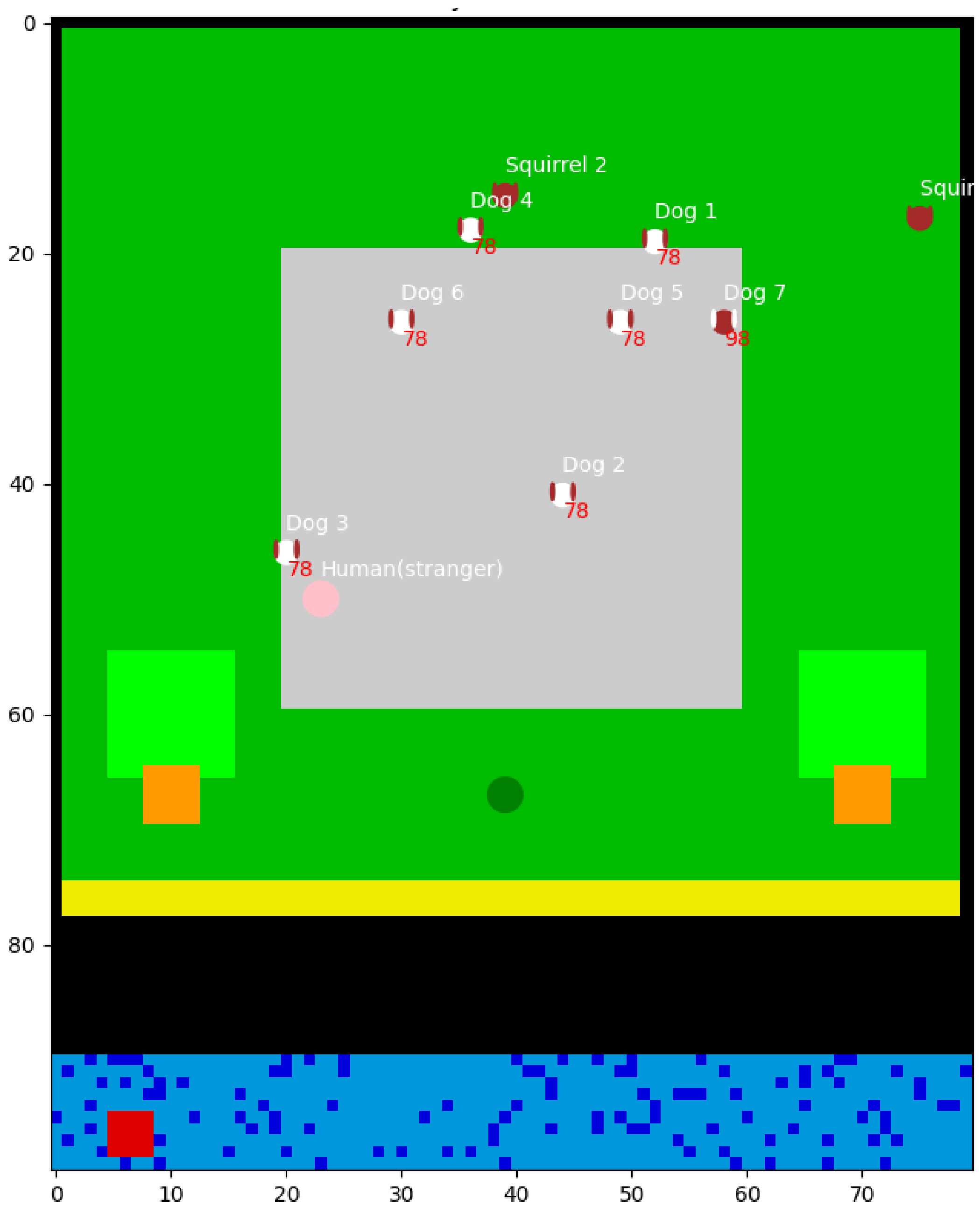

Creatures
- name: str
- colour: strr
- pos: tuple
+ get_pos(): tuple
+ plot_me(ax, LIMITS): None
Puppy
- age: int
- energy: int
+ energy_fading(): None
+ near_humanhumans, ax, LIMITS): None)
+ distance_to_objects(obj): float
+ check_borders(move): tuple
+ reproduce_when_near_puppy(puppies): bool
+ step_change(objs): None
+ bark(ax, LIMITS): None
+ stop_tree_collision(move): tuple
+ is_old(): bool
Squirrel
- age: int
+ distance_to_objects(obj): float
+ check_borders(move): tuple
+ step_change(humanObj, puppyObj): None
Human
- age: int
+ distance_to_objects(obj): float
+ check_borders(move): tuple
+ step_change(): None
+ stop_tree_collision(move): tuple
Ball
1
N
1
1
N
1
N
N

Showcase

To run the simulation, the user needs to enter "python3 assessment.py" command in the terminal. The system will then ask for valid integers for the number of time steps, the number of puppies, the number of squirrels, the number of humans, and the number of balls. Once the inputs are provided, the simulation will start. The area between rows 90 and 100 represents the rainy sky, rows 75 to 78 represent the wall, rows 78 to 90 represent the road, and the grey area represents the house. The rest of the green area is the garden with two trees.

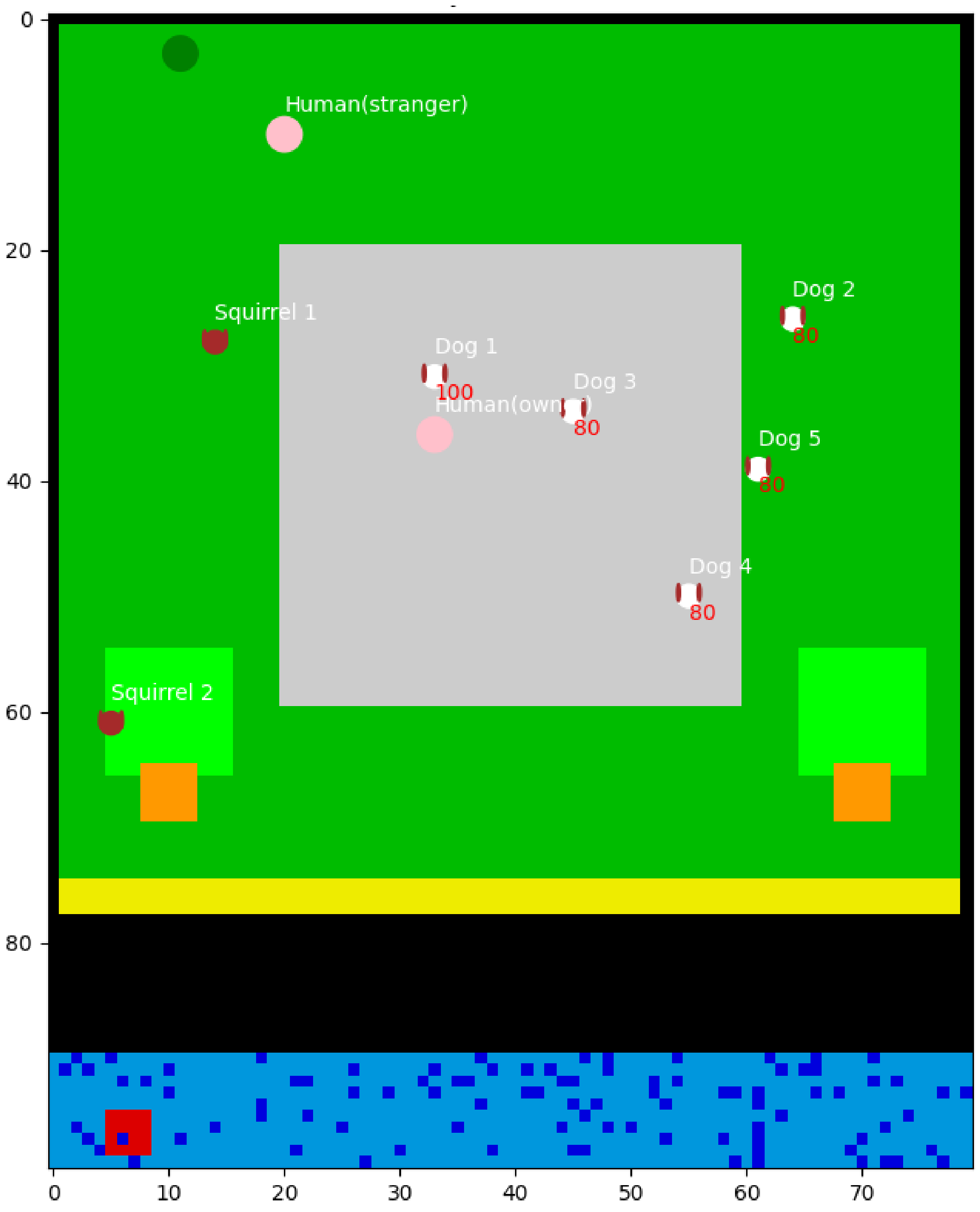
Scenario 1: Reproducing

During the simulation, all individuals move based on their own unique step_change() method. If two adult puppies (dogs aged 10 or older) meet, they have the ability to reproduce and create junior puppies. In the example below, dog 7 is a junior puppy, and its parent puppies are dog 1 and dog 5.



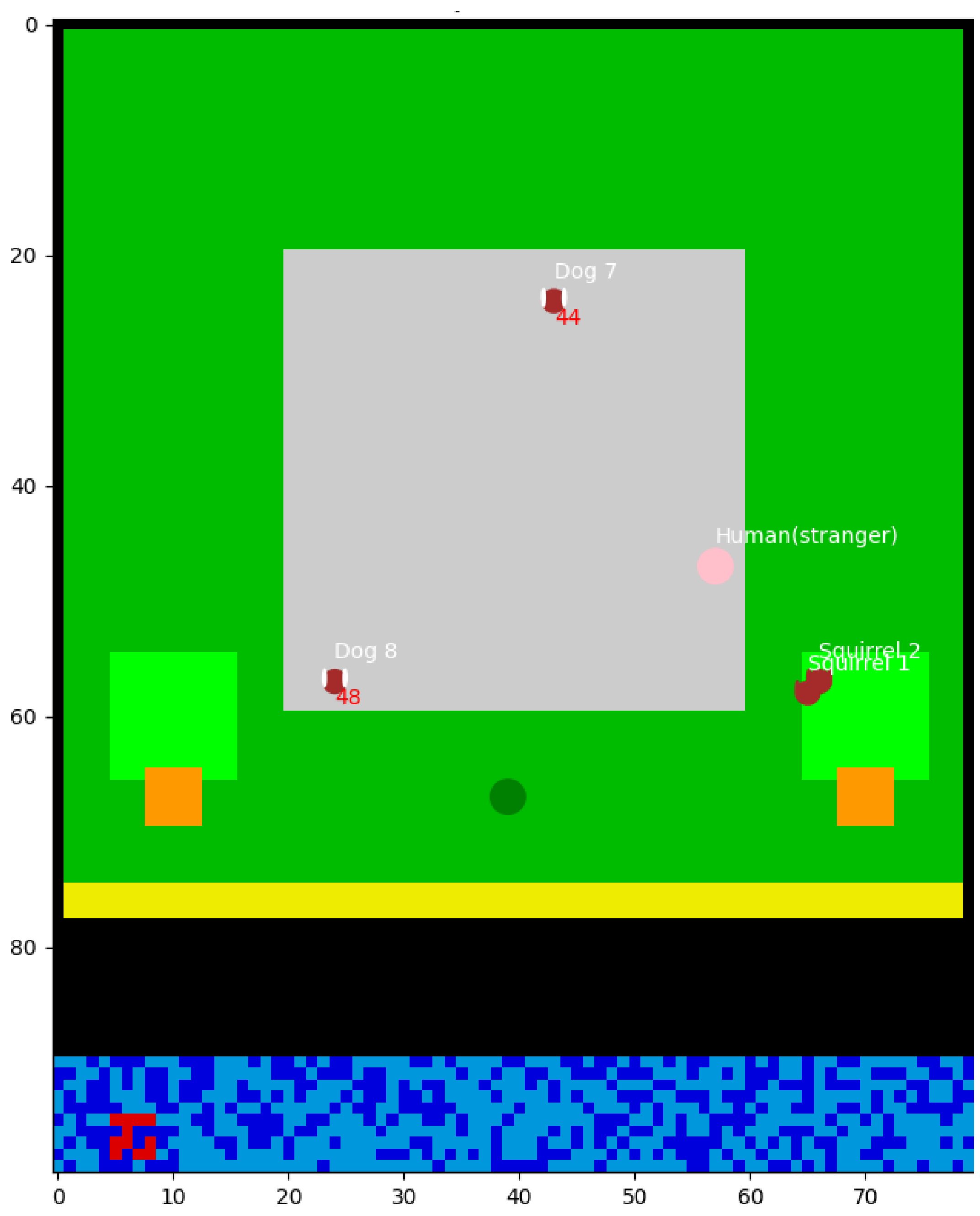
Scenario 2: Feeding puppies

When a puppy approaches its owner, its energy level can increase to 100, indicating that it is being fed or cared for by the owner. In the following scenario, dog 1 has been fed by the owner, causing its energy level to be reset to 100, while the energy levels of the other dogs remain at 80.



Scenario 3: Ending life cycle

The is_age() method is used to demonstrate the life cycle of a puppy by increasing its age in each iteration. In the example below, only junior puppies remain, while all senior puppies have passed away.



Conclusion

The following aspects were addressed in accordance with the provided assignment specifications:

Representations of animals – Puppies, Humans, and Squirrels are objects with attributes for name, colour, position, and age. Functions are used in respective classes to determine their next activities.

Differentiation of humans – Humans were classified based on their relationship with puppies. Owners can pet and feed puppies, which increases puppies' energy, while strangers can make puppies aggressive and bark at them.

Playing with toys – The balls represent toys that puppies can see or smell and play with.

Implementation of senses – In the simulation, all the characters are able to perceive each other's presence through sight or smell. For instance, when a puppy spots a squirrel, it tends to chase after the squirrel, while the squirrel tries to evade the chase. Similarly, when the squirrel notices the presence of a human or a puppy, it modifies its movement to steer clear of them.

Obstacle navigation and collision handling – By utilizing sensing functions in their respective classes, puppies, humans, and squirrels can prevent collisions.

Interactions – When two senior puppies meet, they can reproduce and have new puppies. When a puppy sees its owner, it may get fed, and if it sees a stranger, it may bark.

Life cycle and time progression – In each time step, puppies age and eventually reach their maximum lifespan, at which point they will be removed from the simulator. At the end of the day, any remaining puppies, humans, and squirrels will return to their homes.

Future Work

Future work for this simulation could involve various improvements, such as enhancing visualization through 3D modelling, refining animal behavior algorithms, introducing more complex environmental changes, implementing more interactive human-animal models, integrating machine learning for adaptive behavior, enabling user customization to environments and animals, and optimizing performance for real-time simulation.

References

"Fundamentals of Programming: snoo.py." Accessed April 24th, 2024 via Blackboard COMP1005, 2024

"Matplotlib Documentation — Matplotlib 3.8.4 Documentation." n.d. https://matplotlib.org/stable/index.html.

"The Python Tutorial." n.d. Python Documentation. https://docs.python.org/3/tutorial/index.html.