Project Report : Puppy Simulation

Name:

Student Number:

Date:

# Overview

The purpose of this program is to simulate a dynamic environment where various animals interact with each other and their surroundings. This simulation serves as a simplified model of real-world scenarios, allowing users to observe and analyze the behaviors of different creatures in a controlled setting. Overall, the program provides a platform for studying animal behaviors in various scenarios, enabling users to gain insights into animal interactions and environmental dynamics.

Implemented features of the program include:

1. **Dynamic Environment:** The simulation environment is represented as a grid where animals and objects can move around freely.
2. **Multiple Animal Types:** Different types of animals are implemented, each with unique behaviors and characteristics. These include puppies, squirrels, and humans.
3. **Interactions:** Animals interact with each other and their environment through actions such as chasing squirrel , eating squirrel if successfully caught , puppy recognizing stranger human and becomes friends once human feeds puppy, puppy playing with toy and kicking it away after playing with it , squirrel runs away if senses danger etc.
4. **Toys:** Objects such as toys are introduced into the environment, providing stimuli for animals to interact with.
5. **Simulation Control:** Users can control the simulation parameters such as simulation time and the number of iterations to observe different scenarios and outcomes.

User Guide

**Setup:**

1. Ensure you have Python installed on your system. If not, download and install Python from [python.org](https://www.python.org/).
2. Download the provided code files (**snoo.py** and **animals.py**) and save them in a directory of your choice.

**Required Packages:**

The simulation relies on the following Python packages:

* **numpy**
* **matplotlib**

If you don't have these packages installed, you can install them via pip using the following commands:

- pip install numpy

- pip install matplotlib

**Code Usage:**

1. Open a terminal or command prompt.
2. Navigate to the directory where you saved the code files.
3. Run the **snoo.py** file using Python: -python snoo.py
4. The simulation will start running, and you will see the visualization of the environment and animal interactions.

Traceability Matrix

Table giving overview of features, and the implementation and testing of your code. Example below.

* **Feature** - numbered for easy referencing
* **Code reference** – reference to files/classes/methods or snippets of code only, do not put the whole program in the report OR “Not Implemented”
* **Test reference** – test code or describe how you tested your feature, N/A if not implemented
* **Status** – P = passed tests, S = skipped, F = failed, or N/A
* **Date Completed** – date or “Ongoing” or N/A if not implemented

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Code Reference | Test Reference | Status | Date Completed |
| 1.0 Creatures (example) |  |  |  |  |
| 1.1 Puppy | Puppy class in animals.py | testPuppy.py | P | 4/5/24 |
| Behaviour | plot\_me() method in Puppy class | testPuppy.py – plotting test section | P | 4/5/24 |
| Puppy Movement | step\_change() method in Puppy class | **testPuppy.py** – movement test section | P | 4/5/24 |
| Puppy Interaction with human, squirrel and toy | **step\_change(),playWithToy(), feed(), bark()** method in **Puppy** class | **N/A** | P | 4/5/24 |
| Feed () | **Feed()** method in puppy class | **N/A** | P | 4/5/24 |
| 2.0 Squirrel | **Squirrel** class in **animals.py** | **testSquirrel.py** | P | 4/5/24 |
| 1.2.1 Behaviour | **plot\_me()** method in **Squirrel** class | **testSquirrel.py** – plotting test section | P | 4/5/24 |
| 1.2.2 Squirrel Movement | **step\_change()** method in **Squirrel** class | **testSquirrel.py** – movement test section | P | 7/5/24 |
| 1.2.2.1 Run Away 5 unit if puppy is close | **Step\_change() method in squirrel class.** | **N/A** | P | 7/5/24 |
| 3.0 Human | **Human** class in **animals.py** | **testHuman.py** | P | 7/5/24 |
| Behaviour | **plot\_me()** method in **Human** class | **testHuman.py** – plotting test section | P | 7/5/24 |
| Human Movement | **step\_change()** method in **Human** class | **testHuman.py – movement test section** | P | 7/5/24 |
| Human Interaction | **step\_change()** method in **Human** class | **testHuman.py – movement test section** | P | 5/2/24 |
| Humal attribute (role=owner, stranger) | **Feed\_puppy\_if\_close()** method in Human() class | **N/A** | P | 7/5/24 |
| Feed Puppy | **Feed\_puppy\_if\_close()** method in Human() class | **N/A** | p | 7/5/24 |
| 4.0 Toy | **Toy** class in **animals.py** | **testToy.py** | P | 5/2/24 |
| Behaviour | **plot\_me()** method in **Toy** class | **testToy.py** – plotting test section | P | 7/5/24 |
| Toy Movement | **step\_change()** method in **Toy** class | **testToy.py** – movement test section | P | 7/5/24 |
| Toy Interaction | **step\_change()** method in **Toy** class | N/A | P | 7/5/24 |
| 1. Update\_Smells | **Update\_smells()** method in yards2() function | Practice Test 3 | P | 7/5/24 |
| 1. Day Night Cycle | Defines in the main function with a for loop | N/A | P | 7/5/24 |
| 1. Terrain | Defined in the **build\_yard() & build\_yard()2** class. | N/A | P | 7/5/24 |
| Terrain backyard is inaccessible | Defined in the **build\_yard2()** class. | N/A | P | 7/5/24 |

Discussion

The simulation I have built is like a virtual playground for different creatures and toys. First, I create the playground and add stuff to it like grass and obstacles. Then, I bring in our characters - puppies, humans, squirrels, and toys. Each of these characters has their own name, color, and where they start in the playground. Once everyone's in, the fun begins! The puppies can chase squirrels, play with toys, or even get fed by humans. Squirrels get eaten by puppies if get caught. After puppies are done with playing toys they kick them around. And toys get kicked by other animals if get touched. Humans can feed the puppies if they're nearby and try to pet them. Squirrels scurry around randomly and run away from puppies. To help us see what's happening, I used pictures and colors to show the playground and where everyone is. Finally, I press play, and the simulation runs through different moments, showing us how the characters move and interact.

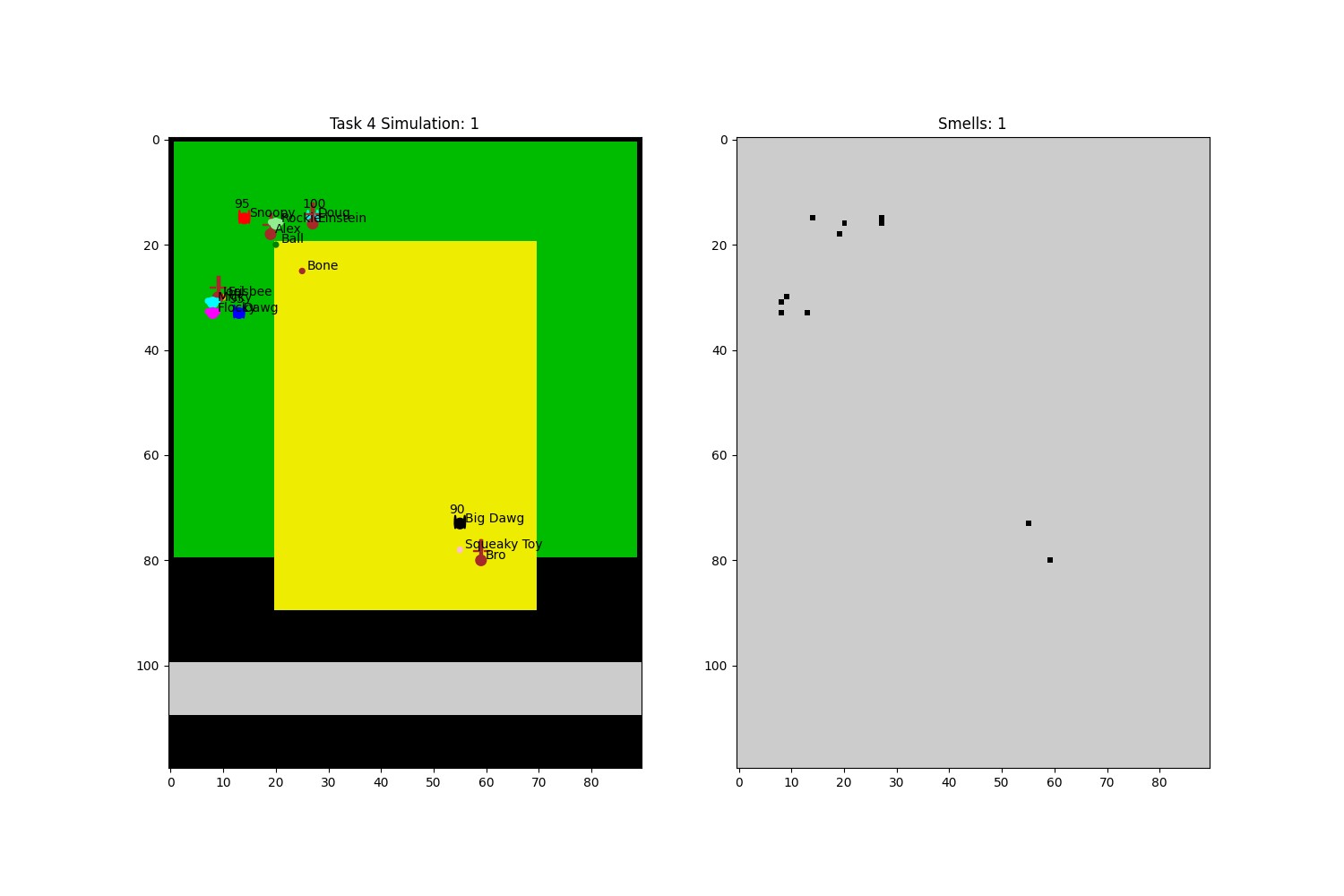
# Showcase

Discussion and demonstration of code output, including three different scenarios – different settings, numbers of items, strategies etc.

In this **Introduction**, describe how you have chosen to set up and compare the simulations for the showcase. Include commands, input files – anything needed to reproduce your results.

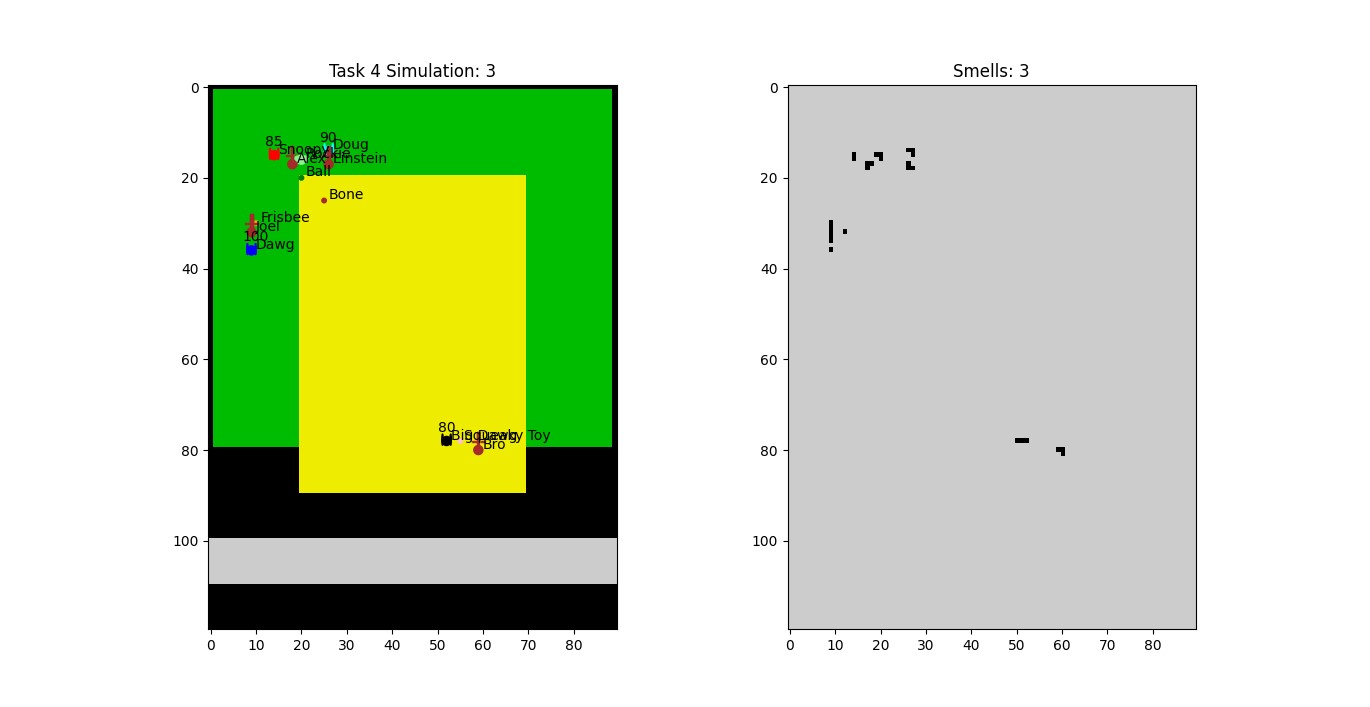
## Scenario 1

## 



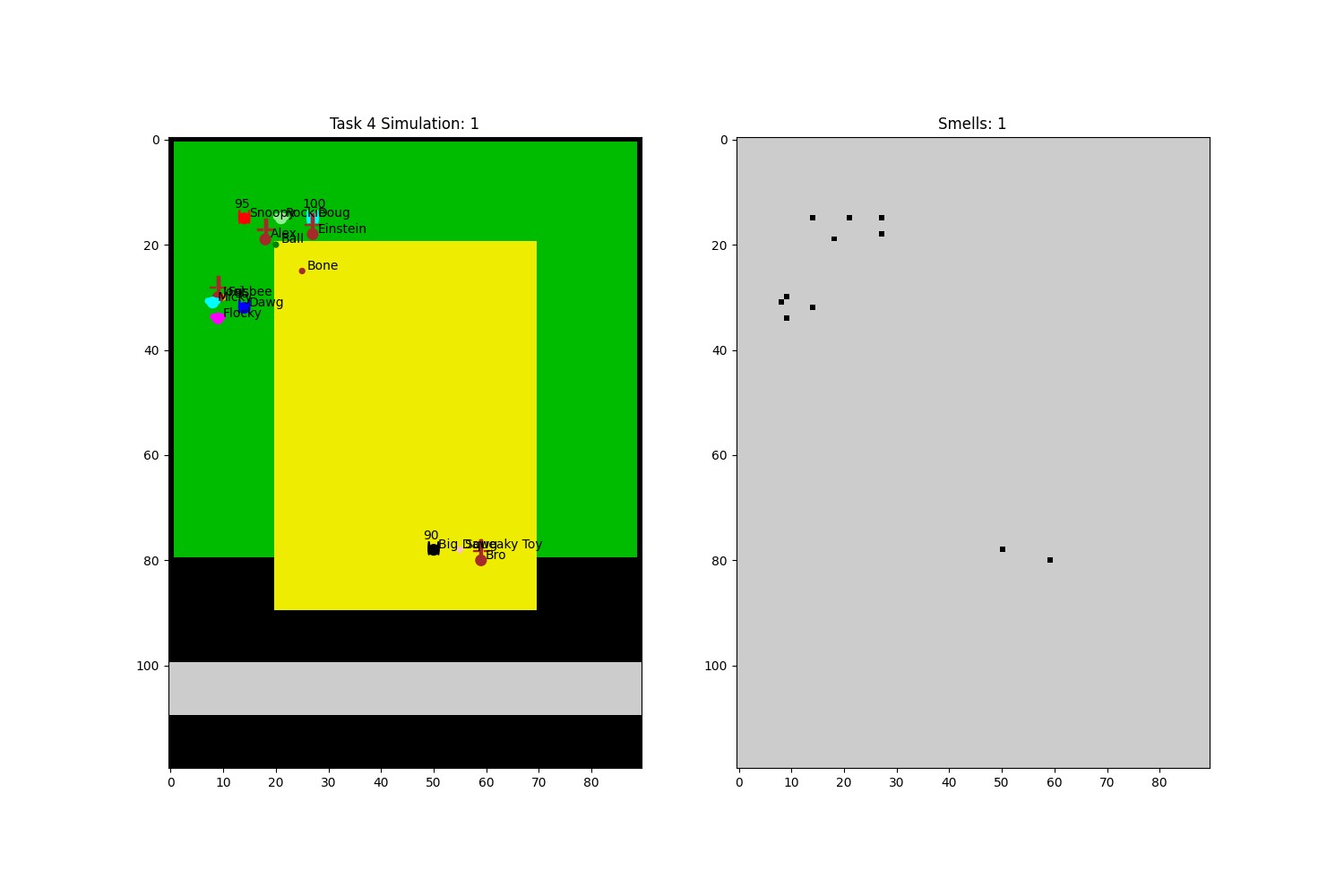
## Scenario 2

## 



## Scenario 3

# 



# Conclusion

In conclusion, the provided code for the Puppy simulation effectively demonstrates the interactions among various creatures in a yard environment. Through the implementation of classes for different creatures like puppies, humans, and squirrels, along with toys, the simulation portrays realistic behaviours such as chasing, playing, and feeding. The code utilizes numpy and matplotlib for yard representation and visualization, enhancing the user experience. Additionally, the simulation incorporates day and night cycles, adding depth to the environment. Overall, the code showcases a structured and engaging simulation framework that effectively fulfills the requirements of the practical test assignment.

# Future Work

There are several avenues for further investigations and extensions that could be explored in the future:

1. **Additional Creature Behaviors**: In future other creature except puppy will also have broad range of behaviour functions.
2. **Environmental Factors**: Type of day, accessibility to terrain and different actions according to different terrain will be introduced in the future.
3. **Genetic Algorithms**: By introducing genetic variation and selection mechanisms, the simulation could simulate the process of natural selection and adaptation in a virtual ecosystem.
4. **Multi-Agent Systems**: Creatures can interact with each other and their environment in a decentralized manner. This could enable behaviors and complex dynamics to arise from simple rules governing individual creature behaviors.
5. **User Interaction**: In future user will be able to interact with the simulation to change or effect many outcomes of the environment .
6. **Integration with Machine Learning**: Integration of machine learning techniques to enable creatures to learn and adapt their behaviors over time through experience and feedback. This could open up new possibilities for creating intelligent and adaptive virtual creatures.

# References

*Here are the references for the technologies and libraries used in the project:*

1. *Python Software Foundation. (2021). Python language reference (Version 3.9) [Computer software]. Python Software Foundation. [Online]. Available:* [*https://www.python.org/*](https://www.python.org/)
2. *Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy [Computer software]. Nature, 585(7825), 357-362. [Online]. Available:* [*https://numpy.org/*](https://numpy.org/)
3. *Hunter, J. D. (2007). Matplotlib: A 2D graphics environment [Computer software]. Computing in Science & Engineering, 9(3), 90-95. [Online]. Available:* [*https://matplotlib.org/*](https://matplotlib.org/)
4. *Amin, Md Samsul. 2024. “PracTest3.” Curtin University. https://mydesktop.curtin.edu.au*