Mahdeen Ahmed Khan Sameer

Professor Bender

CS231

March 2023

Agent-based simulation: A Use of Linked Lists Concept

Abstract:

In our third project, we aim to demonstrate the power of linked lists and object-oriented programming in simulating complex systems, i.e Agent-based Simulations. We will be implementing an agent-based simulation using linked lists on a 2D landscape, starting by wrapping up linked lists, implementing the Iterable interface, and creating an abstract class called Agent. Then, we create the SocialAgent and AntiSocialAgent classes to extend the Agent class, and we add fields and methods to each of these classes to enable them to update their state and move based on their neighbors within a given radius. We also create a Landscape class that stores the agents and implements methods for retrieving the agents within a given radius and visualize the Landscape using LandscapeDisplay. So, we create an AgentSimulation class to control the simulation, generating and randomly placing N agents on the Landscape, and updating their state over time. Overall, this project demonstrates the power of linked lists and object-oriented programming in creating a self-organizing society of agents. Through this simulation, we can explore the behavior of agents under different conditions and gain insights into the emergent properties of complex systems.

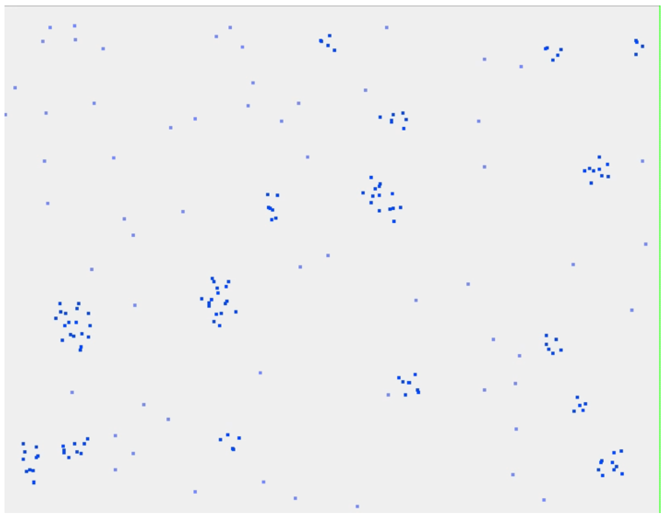
Result:

Throughout this project, I found the instructions to be clear and straightforward. There were multiple ways to approach the implementations, and I had to determine the most precise way to achieve the desired results. However, I did struggle with implementing the getNeighbors() method initially, but with the assistance of the TA's, I was able to use the distance formula correctly to calculate the distance between two points.

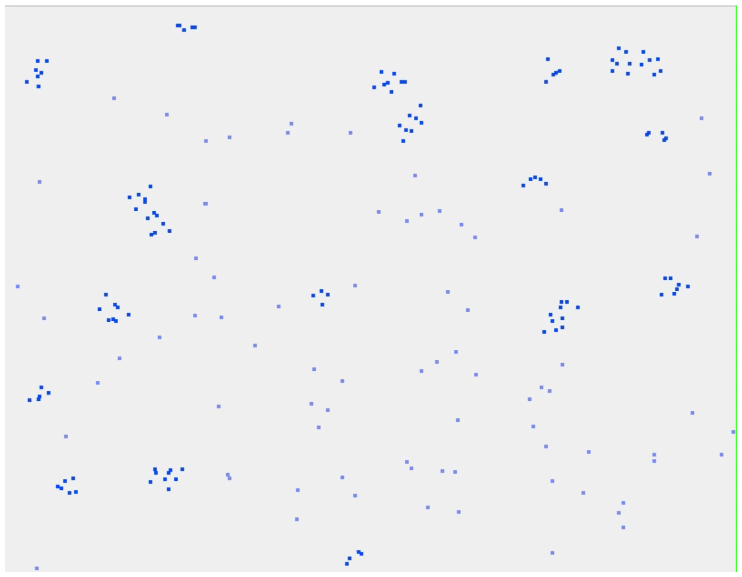
I found the technique of casting data types from int to double, and vice versa, to be very helpful. This technique was particularly useful in avoiding errors that could have occurred from incompatible data types. The TA's advised me to use the "force-casting" method, which proved effective in solving most of the errors I encountered.

Overall, this project was a great opportunity to develop my skills in object-oriented programming and linked lists. I gained insights into the emergent properties of complex systems and learned how to implement and use various data structures to create agent-based simulations. Now, the results for my simulations are as follows.

1. **AgentSimulation() results:**



Video 1: Running AgentSimulation() using Radius = 20



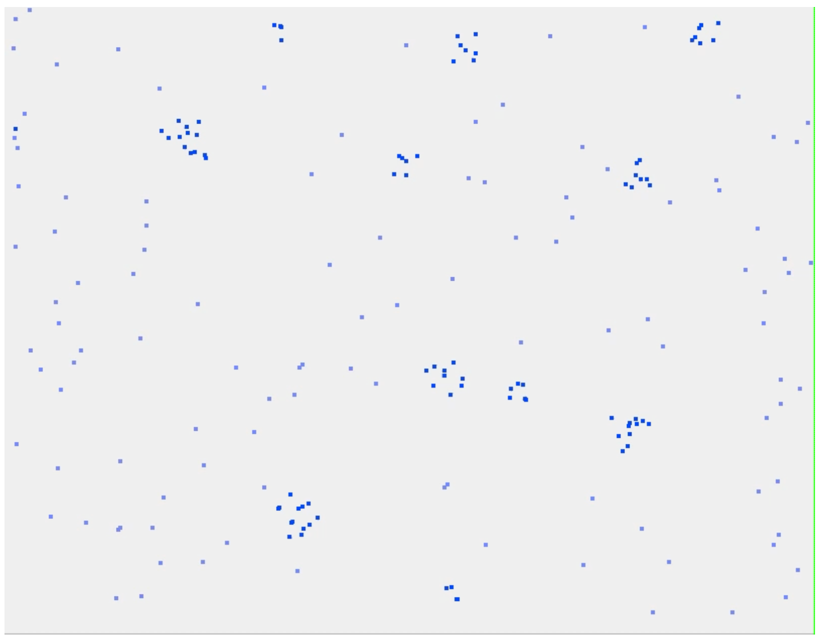
Video 2: Running AgentSimulation() using Radius = 40

To visualize the behavior of agents under different conditions, we tested the simulation with two different radius values: 20 and 40. With a radius of 20, we observed that the agents tended to clump together in small groups, but they did not form a single large group. However, when we increased the radius to 40, we noticed larger and more cohesive groups forming among the agents. This observation highlights how the behavior of agents can be influenced by the radius of sensitivity and how it can affect the overall structure of the system. Through this simulation, we were able to gain insights into the emergent properties of self-organizing societies and the impact of individual behavior on the collective behavior of the system.

Extension 1:

In our extension, we created a new agent class called FearfulAgent that extends the Agent class and includes a radius field that specifies the radius of sensitivity. Thus, we also included a boolean field moved that indicates whether the agent moved during its last updateState call, as well as setRadius and getRadius methods that set and return the radius field, respectively.

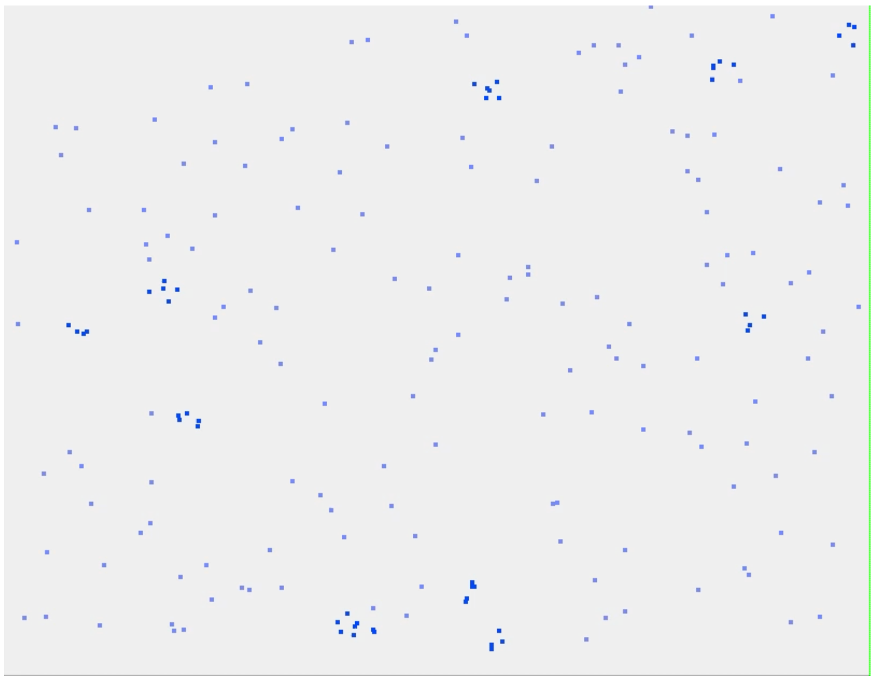
So, to implement the updateState method for FearfulAgent, we used the getNeighbors method of the landscape to find all agents within the agent's radius of sensitivity. We then calculated the distance and direction to each neighbor and moved away from them by updating its x and y coordinates. This behavior of FearfulAgent adds an interesting dimension to the simulation, as it introduces a new type of agent that is programmed to avoid other agents rather than move towards them. This extension allows for more complex interactions between different types of agents and opens new possibilities for exploring emergent behavior in agent-based simulations.



Video 3: Running FearfulAgent()

Extension 2:

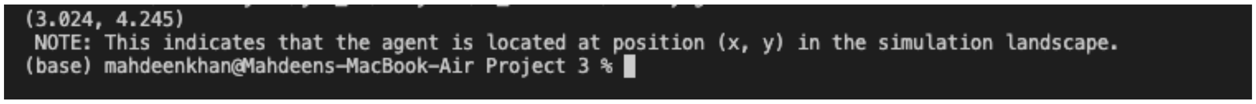
In our second extension, we implemented our agent move towards other agents, or possibly towards some and away from others instead of moving randomly. We did this by creating another class, SocialAgent2, which we implemented the updatestate method to determine the agent's method. We did this by finding the distance and direction to the average position of its neighbors and normalizing a vector to move at a fixed speed. If the new position is within the bounds of the landscape, the agent updates its position to the new position. This implementation allows for the simulation of agents that can move towards or away from others, based on a social parameter. If the social parameter is set to true, the agent moves towards the average position of its neighbors. If the parameter is set to false, the agent moves away from the average position of its neighbors. The updateState method calculates the vector to the average position and normalizes it to move at a fixed speed. The new position is checked to ensure that it is within the bounds of the landscape before updating the agent's position.



Video 4: Running SocialAgent 2

Extension 3:

We also created a child class MyAgent of the abstract class Agent in Java and added new fields and behaviors to it. The MyAgent class has a boolean moved field that indicates whether the agent has moved during its last updateState call and an integer radius field that specifies the agent's radius of sensitivity. The constructor of the MyAgent class initializes the moved and radius fields. The hasMoved and getRadius methods return the values of the moved and radius fields, respectively. So, the main method, we made, creates a MyAgent object with the specified initial position, moved status, and radius, andso prints out the agent's information and a note explaining what the information indicates.



```
(3.024, 4.245)
 NOTE: This indicates that the agent is located at position (x, y) in the simulation landscape.
(base) mahdeenkhan@Mahdeens-MacBook-Air Project 3 %
```

Figure: Running MyAgent class

Extension 4:

In our simulation, we created a Landscape object with dimensions of 500 x 500. We then instantiated a SocialAgent object at position (250, 250) with a radius of 10 and added it to the landscape using the addAgent method. Finally, we created a LandscapeDisplay object using the landscape and called the repaint method to display the simulation.

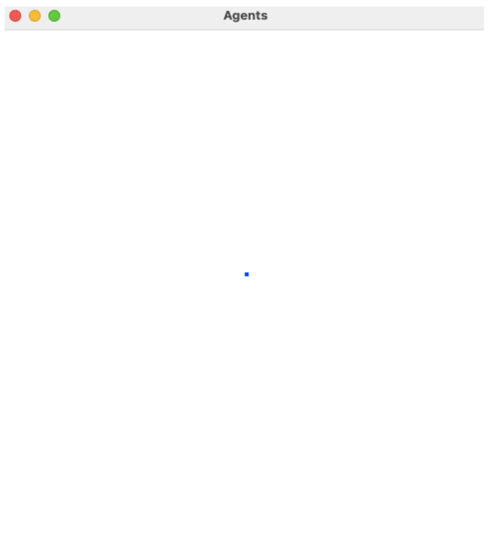


Figure: Running Simulation. POV: Lonely Agent

Reference:

Working on a project is never a solo journey! It requires collaboration, communication, and teamwork to make something great. I had the pleasure of working with a fantastic group of people like Ahmed, Zaynab, Kashef, Nafis on this project, including the amazing teaching assistants Bishal and instructor Professor Max Bender, who provided us with their expertise, guidance, and support. I also want to extend a special thanks to my friend Banshee who have taken the course in a previous semester and shared their insights and experiences with me. His advice was invaluable, and I couldn't have done this without him. I learned so much from them and enjoyed every moment of our collaboration. Above all, thank you so much to our Professor Bender and Dean of Studies to give me an extension for this project, as I went overseas to attend a family wedding, and so managing everything at the same time was my greatest challenge.

Sources, imported libraries, and collaborators are cited:

From my knowledge, there should not be any required sources or collaborators. But we can say, we got our instructions from CS231 Project Manual:

https://cs.colby.edu/courses/S23/cs231B/projects/project1/project1.html

Except this, while working on the classes, I tried to import “java.util.ArrayList”, “java.util.List”, “java.util.awt.*”, “java.util.Scanner”, “java.util.Collections”, “java.util.Random.”