

# CS231 Project 3: Agent-Based Simulation

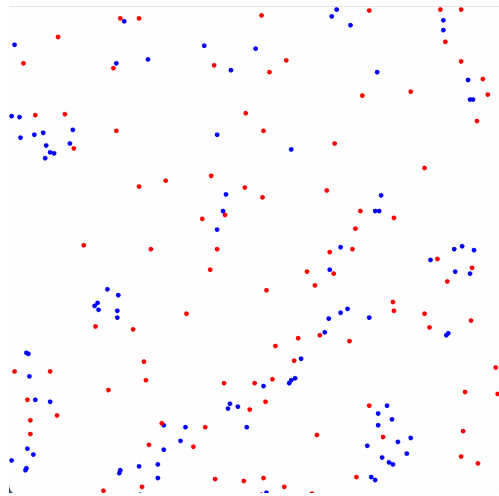
September 26, 2023

## Abstract

This project uses a LinkedList through the context of an agent-based simulation that contains Agents in a 2D landscape. The simulation will contain agents that have positions in a continuous 2D space and an updateState method. Agent-based simulations are used to study interaction between two things, which in this case, how agents behave when given a set of rules. The agents in this simulation include antisocial and social agents. Antisocial agents move away from a group of agents, if there is more than one neighbor within radius, while social agents move away if there are less than four agents within its own radius. At the end of the simulation, the social agents (blue) tend to clump together, while antisocial (red) agents are more spread apart as demonstrated in the results.

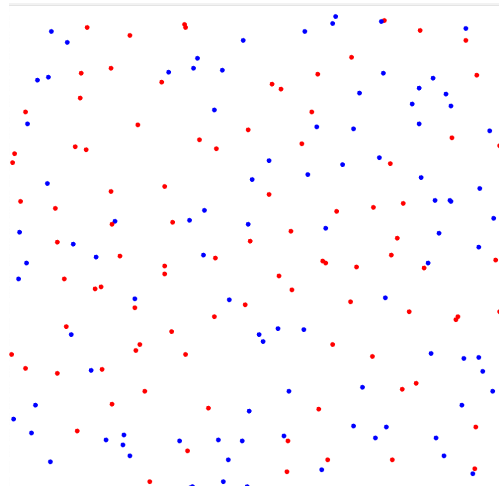
---

## Results



### End of Simulation

For the first simulation that is demonstrated above, the radius for social agents was 25, while the radius for antisocial agents was 15. The number of agents that were added were 200, and they followed a 500 x 500 grid.



### End of Simulation where Radius is twice as big

For the second simulation the radius for the social agents was 50, while the radius for antisocial agents was 30. Therefore, a simulation where the radius was twice as big as the previous simulation. Since the radius is larger compared to the last simulation, there are less agents in clumps and more spread apart as shown in the image. This can be explained since for the AntiSocialAgent, the agent moves if there is one neighbor within its radius while the SocialAgent moves if there is less than 4 within radius, but since radius is too big, barely agents need to move.


## Discussion

The results from the agent-based simulation do make sense, since social agents are supposed to be more clumped together since it moves if there are less than four agents within its radius. On the other hand, antisocial agents move if there is more than one neighbor within its radius, therefore, each agent should be more separate from one another (no social interaction).

---

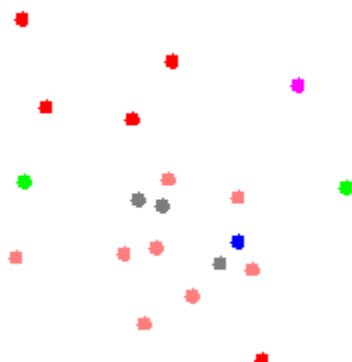
## Extensions

### EnemyAgent



A new type of Agent called the EnemyAgent is similar to the AntiSocialAgent; It will only stay in place if it is the only Agent within its radius, without anyone else near it. Otherwise, it will move. This is marked by the color green on the Landscape Display. However, EnemyAgents are designed to move away from any other Agent that enters its detection radius. It does this by finding the average of the X and Y coordinates of all the Agents within its radius, and measuring the distance between that and itself, it moves in the opposite direction. The EnemyAgent also has a second agent known as the kill radius; If any agent were to somehow get too close, it would die and turn gray, as shown. Dead agents will never move.

### SquadAgent



Another new type of Agent called the SquadAgent which combines both elements of the AntiSocialAgent and Enemy Agent. When spawned, it is marked by the color purple on the Landscape Display. However, unlike all the other Agents, this one will not move whatsoever. It will only activate if an Agent enters its range, to which it disperses into 4 AntiSocialAgents. Afterwards, it will turn gray permanently.

---

## Reflection

### Why did we use LinkedLists instead of, say, Arrays/Arraylists?

For this project, LinkedLists were implemented because there is more efficiency with using them than ArrayLists, which was further explained in my extensions. Additionally, iterators can be implemented through LinkedLists, which simplifies code and helps traverse through a list. For instance, when adding an element to a list at the beginning, compared to arrays, LinkedLists do not

need to make copies, zero temporary memory, and there are three operations regardless of how many items should be added. On the other hand, for ArrayLists, there is a need to create a bigger array, copy all elements to the new array, and then switch references, requiring a longer operation.

---

## **References/Acknowledgements**

I asked Prof. Bender for ideas on extensions as I could not think of anything.

---