CS231: Project 4

Agent-Based Simulations

Parth Parth | CS231L-A | Dr. Alan Harper

Abstract

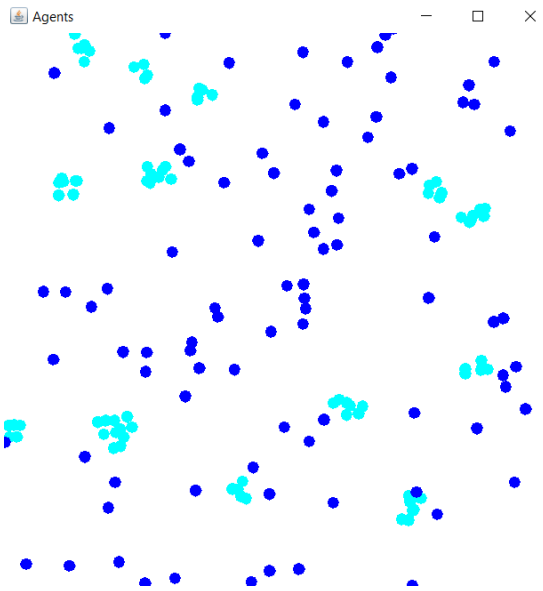
In the project, an agent-based simulation has been implemented. This basically means that a model has been simulated to understand the behaviour and outcomes of autonomous agents, allowing us to see how the behaviour of independent agents and social tendencies affects the outcome for a society.

The data structure that has been used to implement this is a Linked List. A Linked List is a data structure which consists of nodes. A node is a data structure which has (for the purposes of this project) space to store two things in it: a) data and b) a reference to the next node (which can be empty if there is no next node). Using the reference to the first node, all the nodes can be accessed. In this project, the linked list has been used to store all the “agents” that are part of this simulation.

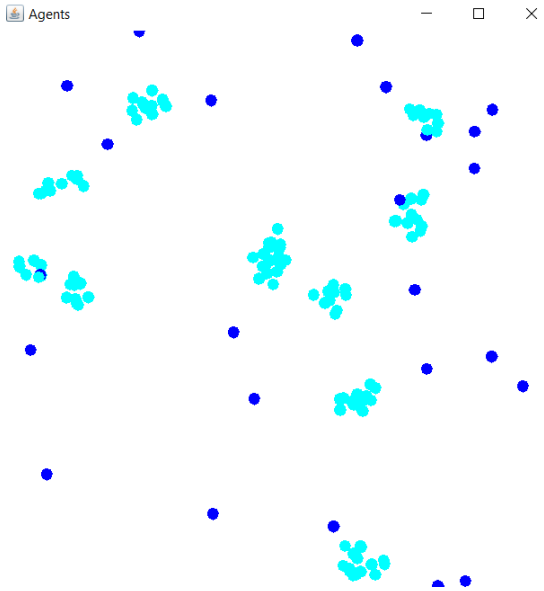
Results

On running the *SocialAgentSimulation* class with radius of 15, it was noticed that after just 100 time steps, the agents had started clumping. At 2000 time steps, there were multiple clumps. The clumps started moving off the canvas after 10,000 time steps and only one or two visible clumps remaining after 20,000 time step

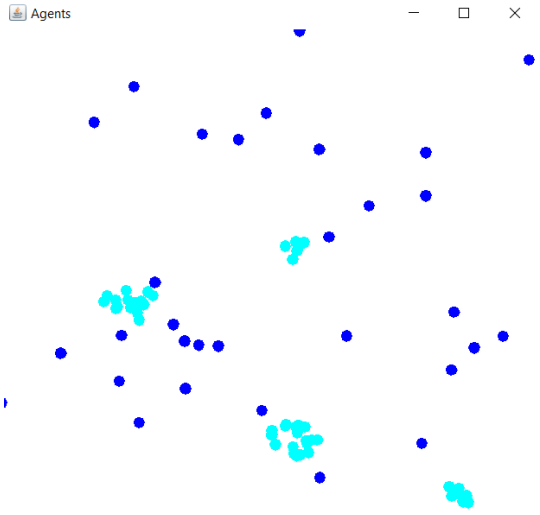
Using a radius of 15



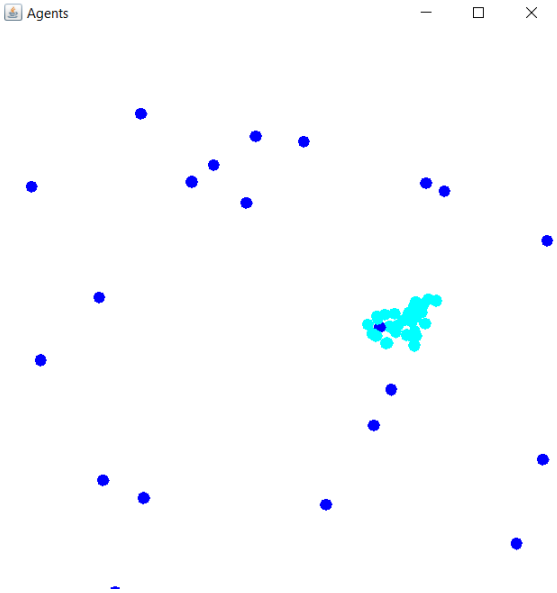
At 100 time steps



At 2,000 time steps



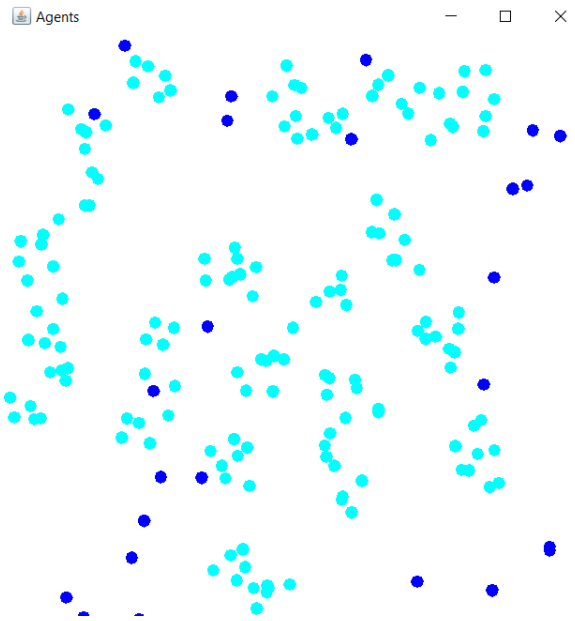
At 10,000 time steps



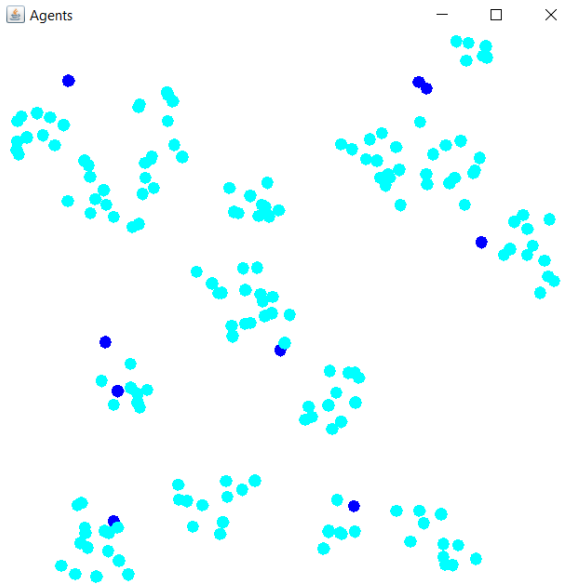
At 20,000 time steps

Using a radius of 30, there was a semblance something happening at 100 time steps (not distinctly clumping). At 2,000 time steps, clumps were visible, although their size was bigger than with radius 15. At 10,000 time steps, the clumps had become more distinct and at 20,000 time steps, there was usually a huge clump with a couple of smaller clumps (as opposed to there being one usually small clump with half the radius).

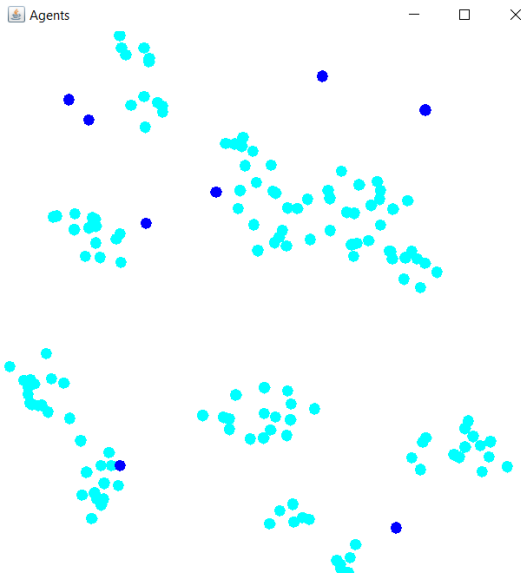
Using a radius of 30



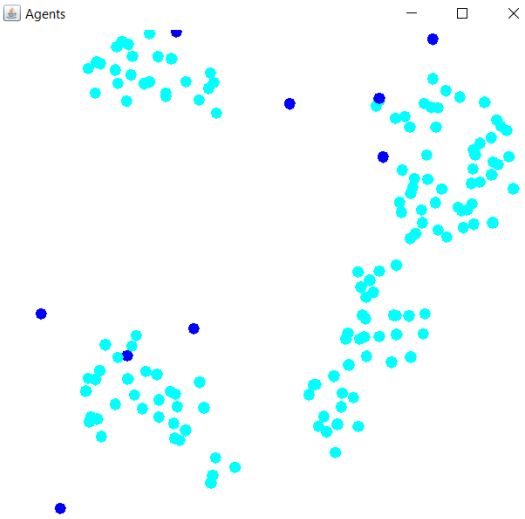
At 100 time steps



At 2,000 time steps



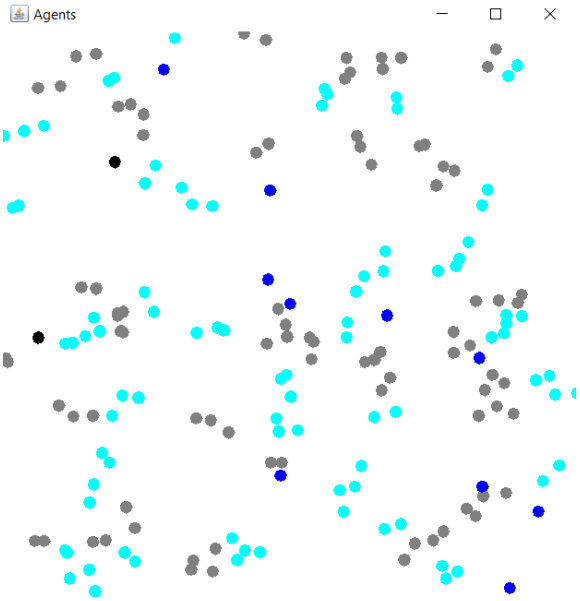
At 10,000 time steps



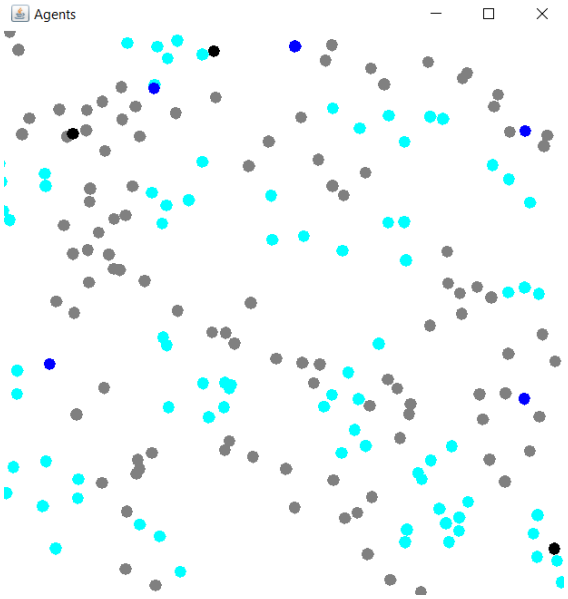
At 20,000 time steps

On running the *CatSocialSimulation* class with 200 time steps, the following results were obtained:

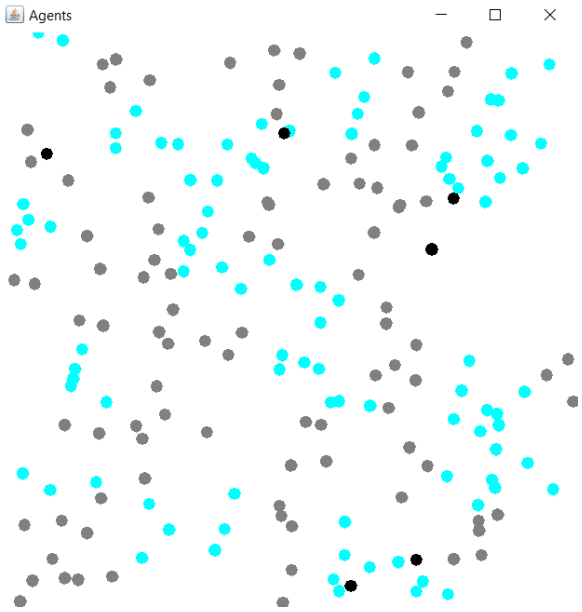
1. Radius=20: Small clumps of categories (in gray and cyan) are vaguely visible. (The moving cells are in blue and black)
2. Radius=40: The clumps are much bigger, but there is still a semblance of clumping in categories.
3. Radius=80: The clumps are very big and look randomly scattered themselves. Without color coding between categories, it would not be wrong to assume that this is a random distribution of agents.



At radius 20



At radius 40



At radius 80

Reflection

To complete the project, I had to learn the concepts of inheritance and method overriding.

From the project, I learnt that agents (which can be analogous to living beings) tend to clump together. The size and number of agents in a clump depends on their radius of sensitivity (which can be thought of as analogous to how social a person is). Agents with a small radius clump together in close groups and agents with bigger radii clump in huge groups that are not as close but have more members.

I also learnt that agents of the same category (maybe analogous to similar interests/motivations) tend to hang out together. Adding radius of sensitivity into this gives the same results, the clumps become huge, but at the cost of closeness/proximity.

Extension 1

For this extension, I have implemented the *ArrayList* class myself. To do so, I am using an array of type *Object*.

The *shuffle()* method has been implemented to use the Fisher-Yates shuffle.

Then, I have made the class iterable by extending the *Iterable<T>* interface.

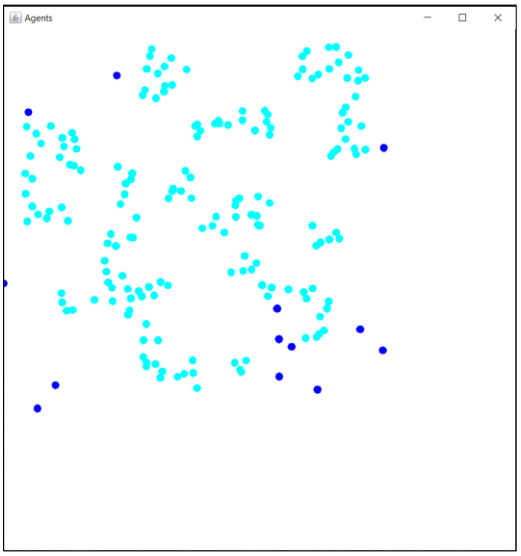
A lot of the challenge of this extension was figuring out exactly how things are done with the *ArrayList* that java provides and replicate those.

Extension 2 (Exploration)

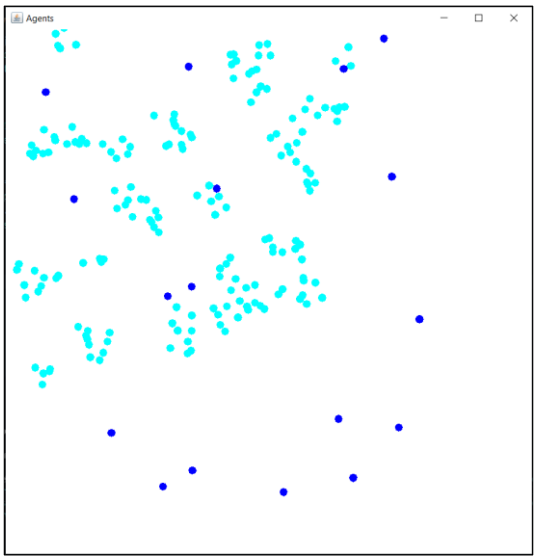
For this extension, I have bound the agents in the canvas to observe the effect it would have on clumping. Previously, the agents were free to move in almost infinite space and would become invisible as they moved off the canvas.

This was achieved by checking for each agent whether it had exceeded the bounds of the canvas and setting it to be at the boundary if it had.

It was observed that the clumps when the agents were bound had more agents in them than when they were not bound. However, the clumps were more pronounced/differentiated when they were not bound.



SocialAgentSimulation with bounded agents



SocialAgentSimulation without bounded agents

References/Acknowledgements

The *LandscapeDisplay* class was retrieved from https://cs.colby.edu/aharper/courses/cs231/f21/labs/lab04/LandscapeDisplay.java . The TA, Isabella Feng helped me debug my LinkedList to fix a bug with my *LinkedList.add(…)* method. Prof. Harper helped my by clarifying what was meant by time steps in the project description.