CS351: Design of Large Programs
Project 4: Disease Simulation

Brooke Chenoweth

Fall 2022

We are going to model a disease spreading through a population. The agents in the simulations could represent humans, rabbits, birds, etc. and the disease could be spread via physical contact, airborne, fluids, etc. We are making a simple simulation where contagious agents expose all agents within a given distance without worrying about the actual details of the disease.

Problem Statement

The agents have at least the following possible states:

- Vulnerable – If exposed to a sick agent, will become sick after some delay
- Sick – Can spread the disease, will either recover or die after some delay
- Immune – Does not get sick after exposure to sick agent
- Dead – Doesn't do anything (is dead)

The transitions between the states may be visualized below.

$$\begin{tikzcd}
\text{Vulnerable} \arrow[dr] & & \\
& \text{Sick} \arrow[dl] \arrow[dr] & \\
\text{Immune} & & \text{Dead}
\end{tikzcd}$$

Initially, most agents start as vulnerable, with a small subpopulation of sick agents to start the spread of the disease. Sick agents notify any agents with some exposure distance that they have been exposed, and after some delay for an incubation period, the vulnerable exposed agents become sick. After some period of sickness, a sick agent will either recover and become immune, or die.

You may prefer to add more states and/or transitions to your simulations. For example

$$\begin{tikzcd}
\text{Vulnerable} \arrow[r, leftrightarrow] & \text{Asympotomatic} \arrow[d] \arrow[ddl] & \\
& \text{Sick} \arrow[ul] \arrow[dl] \arrow[dr] & \\
\text{Immune} \arrow[uu] & & \text{Dead}
\end{tikzcd}$$

Configuration File Format

The configuration file is a plain text file with the configuration options specified one per line, in any order. You decide how to specify which file to use (command line argument, file selection dialog in GUI, etc.) just make sure to document it in your readme so we will know how to use your program. Likewise, if you support additional options beyond those described below, make sure to document them (and perhaps include some example configuration files in your repository to demonstrate the additional behavior).

The meaning of the units in the configuration file is up to you, as long as it is reasonable and documented in your readme file. When it comes to distance and locations, you might choose to interpret it in terms of pixels, but it may work out better to scale the units in your display. The time units for incubation period and sickness time could be thought of as days, but in your simulation, you probably should use some number of hundreds of milliseconds.

I recommend that you just ignore any options that you do not currently support (perhaps with a message to that effect) so that you can more easily share configuration files with each other, even if they contain optional features.

Dimensions

The width and height of the simulation area. Particularly important when initializing agent locations randomly, but may also effect GUI display size.

`dimensions` *`width height`*

If not specified, width and height both default to 200.

Exposure Distance

How close do agents need to be to spread the disease?

`exposuredistance` *`n`*

If not specified, defaults to 20.

Incubation Period

How long after exposure does it take to become sick?

`incubation` *`n`*

If not specified, defaults to 5.

Sickness Time

How long are agents sick before they recover (or die)?

`sickness` *n*

If not specified, defaults to 10.

Recovery Probability

What is the probability that a sick agent will recover (rather than die)?

`recover` *p*

If not specified, defaults to 0.95.

Initial Agent Locations

How many agents are created and where are the agents placed at the beginning of the simulation?

Options are grid, random, and randomgrid (expected to only use one in a given configuration, document in your readme what happens if more than one of these options are specified).

- `grid` *r c* – make a rectangular grid full of agents with r rows and c columns, spaced by exposure distance.
- `random` *n* – make n agents, randomly placed within the simulation area
- `randomgrid` *r c n* – randomly place n agents on a rectangular grid with r rows and c columns

If not specified, defaults to randomly placing 100 agents.

Initial Sick

How many agents are sick at the beginning of the simulation? This many randomly chosen agents will begin the simulation sick.

`initialsick` *n*

If not specified, defaults to 1.

Program GUI

This should display the simulation with different colors based on the changing states of the agents.

You may choose to have the simulation begin immediately upon loading the configuration file, or you might want to add a run button that will start the agents when it is pressed. (As usual, document what we should expect in the readme.)

Implementation Details

- Each agent is an active object running on its own thread.
- Each agent can only talk to its immediate neighbors. (You should not call a method on a neighboring object that calls a method on the next object that calls a method on the next object, and so on.) Pass some sort of message to the neighbor instead and let the neighbor handle the message on its own thread.
- Neighbors are determined via distance. If the agents do not move, you can determine an unchanging set of neighbors during initialization, but if the simulation configuration permits the agent locations to change as the simulation runs, the neighbor set will change dynamically. You may choose to create some sort of shared helper object that has references to all agents to be able to determine the neighbors of a given agent.
- Make sure you are using appropriate concurrent data structures and synchronization to avoid deadlock and race conditions. Suggestion: give every object a BlockingQueue of messages as an "inbox" and just have everything for that object coordinated through processing the messages one at a time.

Additional features

You will get to choose which additional features you want to include in your program.

Whatever additional features you implement, make sure you thoroughly document them in your readme file.

Required

The program should be able to load a configuration file, run the simulation, and display it to the user on a GUI. In addition, you must implement at least 4 of the following additional features.

- Display a history of the simulation in the GUI. This may be as simple as a list of events ("Agent X died at time Y") displayed in a separate pane.
- Display a plot with x axis representing time and y access is number of agents with lines for the various states to graph the progress of the illness. (So you'd see the count of vulnerable agents go down as the number sick go up, and then sick count would decrease as immune and dead go up.)
- Allow agents to move from their initial positions. Enable movement via the configuration file specifying how far and how often they should move. This will complicate determining neighbors within the exposure distance, since they can change dynamically.

  `move` ***`distance delay`*** – Agents should move *distance* amount in a randomly chosen direction every *delay* time units. If not specified, defaults to no agent movement.

- Rerun simulation – allow user to reinitialize and rerun the simulation with the same configuration file settings without having to restart the program. If agents are randomly placed, they'll start in different positions this time, and of course, the exact interactions between the threads will differ, giving a different (though likely rather similar) simulation result.
- Initial immunity – a certain number of agents begin the simulation with immunity (maybe from vaccination?)

  `initialimmune` *`n`*

  If not specified, defaults to 0.
- Additional agent states (perhaps there is a probability of deceased agents rising again as undead? Or, more realistically, perhaps there is a period of asymptomatic contagiousness before being visibly ill.)
- Variable immunity – "immune" agents have some chance of being infected, possibly changing over time as the immune response fades.

Optional ideas

- Implementing more than 4 of the required features listed above.
- Simulation rewind – Record the simulation events and allow the user to rewind/replay the simulation, possibly displaying at a different timescale. (This should *not* be rerunning the simulation, but rather allowing the user to look more closely at the history of the most recently run simulation.)
- Edit simulation configuration in the GUI and save to file in the configuration file format, suitable for loading and running later.
- Multiple diseases – maybe more than one disease is spreading through the population at once? Possibly a mild non-lethal infection grants some immunity to a more deadly disease (like cowpox and smallpox)
- Mitigation policies – reduce probability of vulnerable agents becoming sick after exposure to represent masking, cleaning, air filtration, etc.
- Other disease simulation variations that you can think of. Let me know your ideas!